APPENDIX A

Source code implementing generic effects in a web page using dynamic HTML: snap enclosures on links.

frmBrowser.frm

```
VERSION 5.00
Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.2#0";
"COMCTL32.OCX"
Object = "{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}#1.1#0";
"SHDOCVW.DLL"
Begin VB.Form frmBrowser
   ClientHeight    =   7632
   ClientLeft      =   1548
   ClientTop       =   1332
   ClientWidth     =   9120
   LinkTopic       =   "Form1"
   ScaleHeight     =   7632
   ScaleWidth      =   9120
   ShowInTaskbar   =   0   'False
   Begin VB.Timer timTimer
      Enabled         =   0   'False
      Interval        =   5
      Left            =   7320
      Top             =   1800
   End
   Begin SHDocVwCtl.WebBrowser brwWebBrowser
      Height          =   6864
      Left            =   12
      TabIndex        =   0
      Top             =   25
      Width           =   9684
      ExtentX         =   17082
      ExtentY         =   12107
      ViewMode        =   1
      Offline         =   0
      Silent          =   0
      RegisterAsBrowser=   0
      RegisterAsDropTarget=   1
      AutoArrange     =   -1  'True
      NoClientEdge    =   0   'False
      AlignLeft       =   0   'False
      ViewID          =   "{0057D0E0-3573-11CF-AE69-
08002B2E1262}"
      Location        =   ""
   End
   Begin VB.PictureBox picAddress
      Align           =   1  'Align Top
      BorderStyle     =   0  'None
      Height          =   780
      Left            =   0
      ScaleHeight     =   780
      ScaleWidth      =   9120
      TabIndex        =   1
      TabStop         =   0   'False
      Top             =   0
      Width           =   9120
   End
   Begin ComctlLib.ImageList imlIcons
      Left            =   7080
      Top             =   1000
      _ExtentX        =   804
      _ExtentY        =   804
      BackColor       =   -2147483643
      ImageWidth      =   24
      ImageHeight     =   24
      MaskColor       =   12632256
      _Version        =   327682
      BeginProperty Images {0713E8C2-850A-101B-AFC0-
4210102A8DA7}
         NumListImages   =   8
         BeginProperty ListImage1 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":0000
            Key             =   ""
         EndProperty
         BeginProperty ListImage2 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":0712
            Key             =   ""
         EndProperty
         BeginProperty ListImage3 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":0E24
            Key             =   ""
         EndProperty
         BeginProperty ListImage4 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":1536
            Key             =   ""
         EndProperty
         BeginProperty ListImage5 {0713E8C3-850A 101B AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":1C48
            Key             =   ""
         EndProperty
         BeginProperty ListImage6 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":235A
            Key             =   ""
         EndProperty
         BeginProperty ListImage7 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":2A6C
            Key             =   ""
         EndProperty
         BeginProperty ListImage8 {0713E8C3-850A-101B-AFC0-
4210102A8DA7}
            Picture         =   "frmBrowser.frx":2EC2
            Key             =   ""
         EndProperty
      EndProperty
   End
End
Attribute VB_Name = "frmBrowser"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Public WithEvents IEDocEvents As HTMLDocument
Attribute IEDocEvents.VB_VarHelpID = -1
Public StartingAddress As String
Dim mbDontNavigateNow As Boolean
Dim navigating As Boolean


'*********************************************************
'** Form Loading and Unloading Event Handlers
'*********************************************************
Private Sub Form_Load()
    On Error Resume Next
    ' Play with the address line and beginning navigation
    StartingAddress =
"file://C:\COMDEX\FTWDemo\yahoo\yahoo.html"
    If Len(StartingAddress) > 0 Then
        ' Try to navigate to the starting address
        timTimer.Enabled = True
        brwWebBrowser.Navigate StartingAddress
    End If

    ' Get us visibly ready
    Me.Show
    tbToolBar.Refresh
    Form_Resize

    ' Set up Serra
    Dim result As Long
    result = Serra.CreateSerra(App.hInstance,
frmBrowser.hWnd)
    If (result = 0) Then
        Unload frmBrowser
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Dim result As Long
    result = Serra.DestroySerra()
End Sub

'*********************************************************
'** Web Browser Events
'*********************************************************
Private Sub brwWebBrowser_BeforeNavigate2(ByVal pDisp As
Object, URL As Variant, Flags As Variant, TargetFrameName As
Variant, PostData As Variant, Headers As Variant, Cancel As
Boolean)
    On Error Resume Next
    Set IEDocEvents = Nothing
    Call Serra.StopTouching
    navigating = True
End Sub

Private Sub brwWebBrowser_NavigateComplete2(ByVal pDisp As
Object, URL As Variant)
    ' Play with the address line
    Dim i As Integer
    Dim bFound As Boolean
    Me.Caption = brwWebBrowser.LocationName

    ' Bind the document
    Set IEDocEvents = brwWebBrowser.Document

    navigating = False
End Sub

Private Sub brwWebBrowser_DownloadComplete()
    On Error Resume Next
    Me.Caption = brwWebBrowser.LocationName
End Sub


'*********************************************************
'** IEDocEvents, etc. Handlers
'*********************************************************
'Private Sub IEDocEvents_onmousedown()
'    If brwWebBrowser.Document.parentWindow.event.Button = 4
Then
'        Call Serra.StopTouching
'        Call Serra.StartPushScrolling
'    End If
'End Sub

'Private Sub Form_MouseUp(Button As Integer, Shift As
Integer, X As Single, Y As Single)
 '    Call Serra.StopPushScrolling
 '    Call Serra.StartTouching
'End Sub

'Private Sub IEDocEvents_onmouseup()
'    Call Serra.StopPushScrolling
'    Call Serra.StartTouching
'End Sub

Private Sub IEDocEvents_onmouseover()
    ' The meat of it all!!!
    Dim result As Long
    If (Not navigating) Then
        result = Serra.TryTouching(ByVal
brwWebBrowser.Document.parentWindow.event)
    End If
End Sub

'*******************************************************
'** Address, Toolbar, and Form Event Handlers
'*******************************************************
Private Sub Form_Resize()
    brwWebBrowser.Width = Me.ScaleWidth
    brwWebBrowser.Height = Me.ScaleHeight
End Sub

Private Sub timTimer_Timer()
    If brwWebBrowser.Busy = False Then
        timTimer.Enabled = False
        Me.Caption = brwWebBrowser.LocationName
    Else
        Me.Caption = "Working..."
    End If
End Sub
```

Serra.odl

```
[
uuid(819BD0E0-578E-11d1-A868-0060083A2742),
lcid (0),
helpstring("FeelTheWeb DLL"),
version(0.9)
]

library FeelTheWeb
{
          #define DLLAPI `__stdcall
          importlib("mshtml.dll");
    [
    uuid(819BD0E1-578E-11d1-A868-0060083A2742),
    helpstring("Basic Serra Functions"),
    dllname("FeelTheWeb.dll")
    ]
    module Serra {
                    [
                    entry("CreateSerra"),
                    helpstring("Creates a connection to the
Serra device.  Requires the application's instance handle
and window handle.  Returns 0 if unsuccessful."),
                    ]
                    long DLLAPI CreateSerra( long theHINST,
long theHWND );
                    [
                    entry("DestroySerra"),
                    helpstring("Destroys a connection to the
Serra device.  Returns 0 if unsuccessful."),
                    ]
                    long DLLAPI DestroySerra();
                    [
                    entry("TryTouching"),
                    helpstring("Checks if the given event
touches a touchable element, and if so creates a Serra
enclosure for it."),
                    ]
                    long DLLAPI TryTouching( [in]
IHTMLEventObj* theEventPtr );
                    [
                    entry("StartTouching"),
                    helpstring("Enables Serra's ability to
touch elements."),
                    ]
                    void DLLAPI StartTouching();
                    [
                    entry("StopTouching"),
                    helpstring("Disables Serra's ability to
touch elements."),
                    ]
                    void DLLAPI StopTouching();
                    [
                    entry("StartPushScrolling"),
                    helpstring("Enables the spring effect
for push scrolling."),
                    ]
                    void DLLAPI StartPushScrolling();
                    [
                    entry("StopPushScrolling"),
                    helpstring("Disables the spring effect
for push scrolling."),
                    ]
                    void DLLAPI StopPushScrolling();
    }
}
```

Serra.def

```
LIBRARY    FeelTheWeb

EXPORTS
           CreateSerra
           DestroySerra
           TryTouching
           StartTouching
           StopTouching
           StartPushScrolling
           StopPushScrolling
```

Wrapper.h

```
/*******************************************************
 * FeelTheWeb.dll
 * (c) 1997 Immersion Corporation
 *
 * FILE
 *                  Wrapper.h
 * DESCRIPTION
 *  C++ functions for the FeelTheWeb Visual Basic program.
 *        These are placed in a DLL.
 *        It provides access to MSHTML and the Serra API.
 **/

#ifndef _WRAPPER_H_
#define _WRAPPER_H_
#include "StdAfx.h"
#include "mshtml.h"
#include "vbutil.h"

typedef enum
{
           teCantTouch = 0,
           teAnchor,
           // teArea,?
           teButton,
           teInputButton,
           teinputCheckBox,
           teInputImage,
           teInputText,
           teInputRadio,
           // teMap,?
           teTextArea
}  TouchElem;

// For DLL (public)
long DLLAPI CreateSerra( long theHINST, long theHWND );
long DLLAPI DestroySerra();

long DLLAPI TryTouching( IHTMLEventObj* theEventPtr );

void DLLAPI StartTouching();
void DLLAPI StopTouching();

void DLLAPI StartPushScrolling();
void DLLAPI StopPushScrolling();

// Not For DLL (private)
void _DoEnclosure( IHTMLEventObj* theEventPtr, IHTMLElement*
theElem );
TouchElem _TryTouchingHelper( IHTMLElement* theEl,
IHTMLEventObj* theEventPtr );
TouchElem _CheckIfTouchable(IHTMLElement * theEl);

#endif  _WRAPPER_H_
```

Wrapper.cpp

```
/*******************************************************
 * FeelTheWeb.dll
 * (c) 1997 Immersion Corporation
 * FILE
 *                  Wrapper.cpp
 * DESCRIPTION
 *  C++ functions for the FeelTheWeb Visual Basic program.
 *        These are placed in a DLL.
 *        It provides access to MSHTML and the Serra API.
```

```
*/

#include "StdAfx.h"
#include "comdef.h"
#include <stdio.h>
#include "Wrapper.h"
#include "ForceSerraMouse.h"
#include "ForceSpring.h"
#include "ForceEnclosure.h"
#include "ForcePeriodic.h"

/**********************************************************/
/*  GLOBAL VARIABLES*/
/**********************************************************/
CForceSerraMouse*   gSerraMouse   = NULL;
CForcePeriodic*gEnclosureSnap     = NULL;
CForceEnclosure*gAnchorEnclosure      = NULL;
CForceSpring*gPushSpring     = NULL;

#ifdef DIRECTINPUT_VERSION
          const GUID guidSquare = GUID_Square;
#else
          const GUID guidSquare = GUID_Serra_Square;
#endif

/* Force Effect Parameters */
// Pop Effect
DWORD PDIRX=0, PDIRY=0, PDUR=100, PMAG=2000, PPER=100;
// Enclosure
DWORD ESTIFFH=8000, ESTIFFV=8000, EWWH=8, EWWV=8,
ESATH=10000, ESATV=10000;
// Spring
DWORD SSTIFF=8000, SSAT=10000, SDEAD=5;
// Use pop?
DWORD USEPOP = 1;


/**********************************************************/
/*  PUBLIC FUNCTIONS          */
/**********************************************************/
long DLLAPI CreateSerra( long theHINST, long theHWND )
{
          RECT encRect = { 0, 0, 100, 100 };
          BOOL success;

          // Try to get parameters from FTWfx.dat
          FILE *fp = fopen("FTWfx.dat","r");
          if (fp) {
                    // Pop Effect
                    fscanf(fp,"%d %d %d %d %d", &PDIRX,
&PDIRY, &PDUR, &PMAG; &PPER );
                    // Enclosure
                    fscanf(fp,"%d %d %d %d %d %d", &ESTIFFH,
&ESTIFFV, &EWWH, &EWWV, &ESATH, &ESATV );
                    // Spring
                    fscanf(fp,"%d %d %d", &SSTIFF, &SSAT,
&SDEAD );

                    // Use snap?
                    fscanf(fp,"%d", &USEPOP );
                    // Close it...
                    fclose(fp);
          }

          // Initialize the SerraMouse
    gSerraMouse = new CForceSerraMouse;
          if ( ! gSerraMouse ) goto CS_Err;
    success = gSerraMouse->Initialize( (HINSTANCE)theHINST,
(HWND)theHWND );
          if ( ! success ) goto CS_Err;

    // Create the effects
          gEnclosureSnap = new CForcePeriodic;
          if ( ! gEnclosureSnap ) goto CS_Err;
          success = gEnclosureSnap->Initialize(
                    gSerraMouse,
                    guidSquare,
                    CPoint(PDIRX,PDIRX),    // Direction
                    PDUR,                  // Duration (ms)
                    PMAG,                  // Magnitude
                    PPER                   // Period (ms)
          );
          if ( ! success ) goto CS_Err;

          gAnchorEnclosure = new CForceEnclosure;
          if ( ! gAnchorEnclosure ) goto CS_Err;
          success = gAnchorEnclosure->Initialize(
                    gSerraMouse, // CForceDevice* pDevice,
                    &encRect,     // LPCRECT pRectOutside,
                    ESTIFFH,  // LONG lHorizStiffness,
                    ESTIFFV,   // LONG lVertStiffness,
                    EWWH,     // DWORD dwHorizWallWidth,
                    EWWV,     // DWORD dwVertWallWidth,
                    ESATH,    // DWORD dwHorizSaturation,
                    ESATV,    // DWORD dwVertSaturation,
                    SERRA_FSTIFF_OUTBOUNDANYWALL,
          // DWORD dwStiffnessMask,
                    0x0,        // DWORD dwClippingMask,
                    NULL // CForceCondition*pInsideCondition
    );
          if ( ! success ) goto CS_Err;

          gPushSpring = new CForceSpring;
          if ( ! gPushSpring ) goto CS_Err;
```

```
          success = gPushSpring->Initialize(
                    gSerraMouse,
                    SSTIFF,
                    SSAT,
                    SDEAD,
                    FORCE_EFFECT_AXIS_BOTH,
                    FORCE_SPRING_DEFAULT_CENTER_POINT
          );
          if ( ! success ) goto CS_Err;

          return 1;
CS_Err:
          // We had problems... clean up and leave.
          DestroySerra();
          return 0;
}

long DLLAPI DestroySerra()
{
          if ( gSerraMouse )  { delete gSerraMouse;
gSerraMouse = NULL; }
          if ( gEnclosureSnap)  { gEnclosureSnap->Stop();
delete gEnclosureSnap;          gEnclosureSnap      = NULL; }
          if ( gAnchorEnclosure )  { gAnchorEnclosure-
>Stop(); delete gAnchorEnclosure;   gAnchorEnclosure= NULL;
}
          if ( gPushSpring )  { gPushSpring->Stop();
delete gPushSpring;             gPushSpring          = NULL; }
return 1;
}


long DLLAPI TryTouching( IHTMLEventObj* theEventPtr )
{
          long                  result = 0;
          IHTMLElement*         pEl;
          TouchElem             te;

          // Get the srcElement
          theEventPtr->get_srcElement( &pEl );
          if ( pEl )
          {
              // Check if its touchable
              te = _TryTouchingHelper( pEl, theEventPtr );
              if ( te != teCantTouch )
              {
                    //Create enclosure for element...
                    _DoEnclosure( theEventPtr, pEl );
                    result = 1;
              }
              pEl->Release();
          }
          return result;
}

void DLLAPI StartTouching()
{
          if ( gAnchorEnclosure ) gAnchorEnclosure->Start();
}


void DLLAPI StopTouching()
{
    if ( gAnchorEnclosure )   gAnchorEnclosure->Stop();
          if ( gEnclosureSnap   ) gEnclosureSnap->Stop();
}


void DLLAPI StartPushScrolling()
{
          if ( gPushSpring ) gPushSpring->Start();
}


void DLLAPI StopPushScrolling()
{
          if ( gPushSpring ) gPushSpring->Stop();
}


/**********************************************************/
/*  PRIVATE FUNCTIONS          */
/*********************************************************/
TouchElem _TryTouchingHelper( IHTMLElement* theEl,
IHTMLEventObj* theEventPtr )
{
          IHTMLElement*       pEl;
          TouchElem           te;

          // Our base case
          if ( theEl == NULL )  return teCantTouch;

          // Is this guy touchable?
          te = _CheckIfTouchable( theEl );
          if ( te != teCantTouch )
          {
                    return te;
          }

          //not touchable, try his parent! (if he has one)
          theEl->get_parentElement( &pEl );
          if ( pEl != NULL )
          {
```

```
                        te = _TryTouchingHelper( pEl,
theEventPtr );
                        pEl->Release();
                        return te;
            }
            return teCantTouch;
}


/*  _DoEnclosure
 *  Input:    IHTMLEventObj*, IHTMLElement*
 *  returns: void
 *  Given an event object and an element, creates an
enclosure
 *  for that element.  The event object is for ascertaining
the screen coordinates of the element.
 */
void _DoEnclosure( IHTMLEventObj* theEvent, IHTMLElement*
theElem )
{
            RECT        r;
            long        temp;

            // Process left
            theElem->get_offsetLeft( &(r.left) );
            theEvent->get_screenX( &temp );  r.left += temp;
            theEvent->get_offsetX( &temp );  r.left -= temp;

            // Process top
            theElem->get_offsetTop(  &(r.top) );
            theEvent->get_screenY( &temp );  r.top += temp;
            theEvent->get_offsetY( &temp );  r.top -= temp;

            // Process right and bottom
            theElem->get_offsetWidth(  &(r.right) );
            r.right  += r.left;
            theElem->get_offsetHeight( &(r.bottom) );
            r.bottom += r.top;

            // Calculate wall widths and heights
            DWORD hww = (r.right-r.left+1)/2;
            DWORD vww = (r.bottom-r.top+1)/2;
            if ( hww < EWWH )
                        hww--;
            else
                        hww = EWWH;
            if ( vww < EWWV )
                        vww--;
            else
                        vww = EWWV;

            // Make the enclosure
            if (gAnchorEnclosure)
            {
                        gAnchorEnclosure->ChangeParameters(
                                    &r,
                                    FORCE_EFFECT_DONT_CHANGE,
                                    FORCE_EFFECT_DONT_CHANGE,
                                    hww,
                                    vww,
                                    FORCE_EFFECT_DONT_CHANGE,
                                    FORCE_EFFECT_DONT_CHANGE,
                                    FORCE_EFFECT_DONT_CHANGE,
                                    FORCE_EFFECT_DONT_CHANGE,
                                    (CForceEffect*)
FORCE_EFFECT_DONT_CHANGE
                  );
                        gAnchorEnclosure->Start();
                        if ( USEPOP )
                        {
                                    gEnclosureSnap->Start();
//                                  gEnclosureSnap->Stop();
                        }
            }
}


TouchElem _CheckIfTouchable(IHTMLElement * theEl)
{
            IHTMLElement*        pUnk;

            // Is it an Anchor?
            theEl->QueryInterface( IID_IHTMLAnchorElement,
(LPVOID*)&pUnk );
            if ( pUnk )  {  pUnk->Release();  return teAnchor;
}

            // teArea,?
            // teMap,?

            // Is it a Text Area?
            theEl->QueryInterface( IID_IHTMLTextAreaElement,
(LPVOID*)&pUnk );
            if ( pUnk )  {  pUnk->Release();  return
teTextArea;  }

            // Is it a Button?
            theEl->QueryInterface( IID_IHTMLButtonElement,
(LPVOID*)&pUnk );
            if ( pUnk )  {  pUnk->Release();  return teButton;
}

            // Is it an Input Button?
            theEl->QueryInterface(
IID_IHTMLInputButtonElement, (LPVOID*)&pUnk );
            if ( pUnk )  {  pUnk->Release();  return
teInputButton;  }

            // Is it an Input Check Box?
//          theEl->QueryInterface(
IID_IHTMLInputCheckBoxElement, (LPVOID*)&pUnk );
//          if ( pUnk )  {  pUnk->Release();  return
teInputCheckBox;  }

            // Is it an Input Image?
//          theEl->QueryInterface( IID_IHTMLInputImageElement,
(LPVOID*)&pUnk );
//          if ( pUnk )  {  pUnk->Release();  return
teInputImage;  }

            // Is it an Input Text?
            theEl->QueryInterface( IID_IHTMLInputTextElement,
(LPVOID*)&pUnk );
            if ( pUnk )  {  pUnk->Release();  return
teInputText;  }

            // Is it a Input Radio Button?
            theEl->QueryInterface(
IID_IHTMLOptionButtonElement, (LPVOID*)&pUnk );
            if ( pUnk )  {  pUnk->Release();  return
teInputRadio;  }

            // None of the above!
            return teCantTouch;
}
```

TouchCheck.h

```
// TouchCheck.h: interface for the CTouchCheck class.
//
//////////////////////////////////////////////////////////////////////

#if
!defined(AFX_TOUCHCHECK_H__E8568F20_4BAC_11D1_A868_0060083A2
742__INCLUDED_)
#define
AFX_TOUCHCHECK_H__E8568F20_4BAC_11D1_A868_0060083A2742__INCL
UDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "StdAfx.h"
#include <afxtempl.h>
#include "mshtml.h"

typedef enum
{
            teCantTouch = 0,
            teAnchor,
            // teArea,?
            teButton,
            teInputButton,
            teinputCheckBox,
            teInputImage,
            teInputText,
            teInputRadio,
            // teMap,?
            teTextArea
}  TouchElem;

typedef struct
{
            RECT        frame;
            TouchElem kind;
}  TRECT;


class CTouchCheck
{
public:
            static long p_mTouchProtected;
            void SetScreenToClient( long Xval, long Yval );
            void SetScrollVal( long left, long top );
            void SetClientRect( long left, long top, long
right, long bottom );

            int TryTouching( long mX, long mY );

            void FillTouchables( IHTMLElementCollection*
theAll );

            static TouchElem CheckIfTouchable( IHTMLElement*
theEl);

            static void EnableTouching( int touching );
            static long IsTouchingEnabled();
            static long IsReadyToTouch();

            CTouchCheck();
```

```
		virtual -CTouchCheck();

protected:
		CArray<TRECT, TRECT&> *p_mTouchables;
		RECT p_mClientRect;
		int p_mTouchableSize;
		int p_mCurrentIndex;
		static long p_mReadyToTouch;		//
Internally Controlled
		static long p_mTouchingEnabled;
		// User Controlled
		int p_mClientRectSpecified;
		long p_mScrollLeft;
		long p_mScrollTop;
		long p_mScreenToClientX;
		long p_mScreenToClientY;

private:
		void _GetTouchableFrame( IHTMLElement* theEl,
RECT* theRect);
		void _TransformFrameCorner( IHTMLElement* theEl,
long* left, long* top );
		static void _copyRect( RECT* from, RECT* to );
		void _copyRectWithOffset( RECT* from, RECT* to );
		static int _outside( RECT* theRect, long theX,
long theY);
		static int _inside( RECT* theRect, long theX, long
theY);
		void CTouchCheck::_clipRectToClientRect( RECT* r
);
};

#endif //
!defined(AFX_TOUCHCHECK_H__E8568F20_4BAC_11D1_A868_0060083A2
742__INCLUDED_)


------------------------------------------------------------
```

TouchCheck.cpp

```
// TouchCheck.cpp: implementation of the CTouchCheck class.
//
//////////////////////////////////////////////////////////////

#include "TouchCheck.h"
//#include <winbase.h>
#include <comdef.h>
//#include "OutData.h"
//#include "FeedBack.h"

//extern  CFeedBack*		g_pFdBk;
RECT				g_rcObj;

// Static Members
long CTouchCheck::p_mTouchingEnabled = 0;
long CTouchCheck::p_mReadyToTouch = 0;
long CTouchCheck::p_mTouchProtected = 0;

// Macro
#define _ResetReadyToTouch()	InterlockedExchange(
&p_mReadyToTouch, 0 )
#define _SetReadyToTouch()	InterlockedExchange(
&p_mReadyToTouch, 1 )
#define _ResetTouchingEnabled()		InterlockedExchange(
&p_mTouchingEnabled, 0 )
#define _SetTouchingEnabled() InterlockedExchange(
&p_mTouchingEnabled, 1 )
#define _ResetTouchProtected()		InterlockedExchange(
&p_mTouchProtected, 0 )
#define _SetTouchProtected()	InterlockedExchange(
&p_mTouchProtected, 1 )


//////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////
CTouchCheck::CTouchCheck()
{
		p_mTouchables = new CArray<TRECT, TRECT&>;
		p_mTouchableSize = -1;
		p_mCurrentIndex  = -1;
		p_mClientRectSpecified = 0;
		_ResetReadyToTouch();
		_ResetTouchingEnabled();
}

CTouchCheck::-CTouchCheck()
{
		_ResetReadyToTouch();
		_ResetTouchingEnabled();
		while ( p_mTouchProtected == 1 ) Sleep(0);
		// Yield until resources are free
		delete p_mTouchables;
}

/************************************************************/
/**  Public Functions
/************************************************************/
void CTouchCheck::FillTouchables(IHTMLElementCollection *
theAll)
{
		HRESULT				hr;
		TouchElem		kind;
		long				index, allLength, i;
		VARIANT				vIndex, var2;
		LPDISPATCH			pDisp;
		IHTMLElement*		pElem = NULL;
//		IHTMLElementCollection* pAll = NULL;
		TRECT				theTRect;

		// Clear out the previous touchables!
		_ResetReadyToTouch();
		p_mTouchables->RemoveAll();

		// Grab our document.all interface...
//		hr = theAll->QueryInterface(
IID_IHTMLElementCollection, (LPVOID*)&pAll );
//		if ( hr == S_OK )
//		{
		// Go through the list and onlykeep touchable ones
				index = -1;
				vIndex.vt = VT_UINT;
				VariantInit( &var2 );
				theAll->get_length( &allLength );
				p_mTouchables->SetSize( allLength );
				for ( i=0; i < allLength; i++ )
				{
						// Get the element
						vIndex.lVal = i;
						hr = theAll->item( vIndex,
var2, &pDisp );
						if ( hr == S_OK )
						{
								hr = pDisp-
>QueryInterface( IID_IHTMLElement, (LPVOID*)&pElem );
								if ( hr == S_OK )
								{
									// See if it's
touchable, and if so, add it to the array
									kind =
CheckIfTouchable(pElem);
									if ( kind !=
teCantTouch )
									{
		index++;
		theTRect.kind = kind;
		_GetTouchableFrame( pElem, &(theTRect.frame) );
		p_mTouchables->SetAt( index, theTRect );
									}
									pElem-
>Release();
								}
								pDisp->Release();
						}
				}
//		}
		p_mTouchableSize = index;
		p_mCurrentIndex  * -1;
		_SetReadyToTouch();
}


// mX and mY are in screen coordinates
int CTouchCheck::TryTouching(long mX, long mY)
{
		// Check if we're ready for action!
		if ( ! (CTouchCheck::p_mReadyToTouch &&
p_mClientRectSpecified && CTouchCheck::p_mTouchingEnabled) )
				return 0;

		// Stop immediately if out of bounds...
		if ( _outside( &p_mClientRect, mX, mY ) )
				return 0;

		// Transform mouse coordinates into client
coordinates
		mX -= (p_mScreenToClientX - p_mScrollLeft);
		mY -= (p_mScreenToClientY - p_mScrollTop );

		// Check everything in client coordinates!
		int				index = 0;
		TRECT		aTR;
		while ( index <= p_mTouchableSize )
		{
				aTR = p_mTouchables->GetAt(index);
				if ( _inside( &(aTR.frame), mX, mY ) )
				{
						// Is it the same one we're
currently touching?
						if ( index == p_mCurrentIndex)
								return 0;

						// Do Feedback on it... maybe
later check its touchtype (kind)
						_copyRectWithOffset(
&(aTR.frame), &(g_rcObj) );
						_clipRectToClientRect(
&(g_rcObj) );
//						g_pFdBk->Enable( FBID_TREEITEM
);
						p_mCurrentIndex = index;
						return 1;
				}
```

```
                        index++;
                }
                return 0;
}


void CTouchCheck::SetClientRect(long left, long top, long
right, long bottom)
{
                p_mClientRect.left    = left;
                p_mClientRect.top     = top;
                p_mClientRect.right   = right;
                p_mClientRect.bottom = bottom;
                p_mClientRectSpecified = 1;
}


void CTouchCheck::SetScrollVal(long left, long top)
{
                _ResetReadyToTouch();
                p_mScrollLeft = left;
                p_mScrollTop  = top;
                p_mCurrentIndex  = -1;
                _SetReadyToTouch();
}


void CTouchCheck::SetScreenToClient(long Xval, long Yval)
{
                p_mScreenToClientX = Xval;
                p_mScreenToClientY = Yval;
}

void CTouchCheck::EnableTouching( int touching )
{
                if (touching)
                                _SetTouchingEnabled();
                else
                                _ResetTouchingEnabled();
}

long CTouchCheck::IsTouchingEnabled()
{
                return CTouchCheck::p_mTouchingEnabled;
}

long CTouchCheck::IsReadyToTouch()
{
                return (CTouchCheck::p_mReadyToTouch &&
CTouchCheck::p_mTouchingEnabled);
}


/***********************************************************/
/**  Private Functions
/***********************************************************/
void CTouchCheck::_GetTouchableFrame(IHTMLElement * theEl,
RECT * theRect)
{
//              long            left, top, right, bottom;

                theEl->get_offsetLeft( &(theRect->left) );
                theEl->get_offsetTop( &(theRect->top) );
                theEl->get_offsetWidth( &(theRect->right) );
                theEl->get_offsetHeight( &(theRect->bottom) );

                _TransformFrameCorner( theEl, &(theRect->left),
&(theRect->top) );

                theRect->right  += (theRect->left);
                theRect->bottom += (theRect->top);
}

void CTouchCheck::_TransformFrameCorner( IHTMLElement*
theEl, long* left, long* top )
{
                long                            temp;
                IHTMLElement*           pEl;

                theEl->get_offsetParent((IHTMLElement**)&pEl);
                if ( pEl != NULL )
                {
                                pEl->get_offsetLeft( &temp );
                                *(left) += temp;
                                pEl->get_offsetTop( &temp );
                                *(top)  += temp;

                                _TransformFrameCorner( pEl, left, top );
                                pEl->Release();
                }
}

TouchElem CTouchCheck::CheckIfTouchable(IHTMLElement *
theEl)
{
                IHTMLElement*           pUnk;
                // Is it an Anchor?
                theEl->QueryInterface( IID_IHTMLAnchorElement,
(LPVOID*)&pUnk );
                if ( pUnk )  {  pUnk->Release();  return teAnchor;
}
                // teArea,?
                // teMap,?
                // Is it a Text Area?
                theEl->QueryInterface( IID_IHTMLTextAreaElement,
(LPVOID*)&pUnk );
                if ( pUnk )  {  pUnk->Release();  return
teTextArea;  }

                // Is it a Button?
                theEl->QueryInterface( IID_IHTMLButtonElement,
(LPVOID*)&pUnk );
                if ( pUnk )  {  pUnk->Release();  return teButton;
}

                // Is it an Input Button?
                theEl->QueryInterface(
IID_IHTMLInputButtonElement, (LPVOID*)&pUnk );
                if ( pUnk )  {  pUnk->Release();  return
teInputButton;  }

                //              // Is it an Input Check Box?
//              theEl->QueryInterface(
IID_IHTMLInputCheckBoxElement, (LPVOID*)&pUnk );
//              if ( pUnk )  {  pUnk->Release();  return
teInputCheckBox;  }

                //              // Is it an Input Image?
//              theEl->QueryInterface( IID_IHTMLInputImageElement,
(LPVOID*)&pUnk );
//              if ( pUnk )  {  pUnk->Release();  return
teInputImage;  }

                // Is it an Input Text?
                theEl->QueryInterface( IID_IHTMLInputTextElement,
(LPVOID*)&pUnk );
                if ( pUnk )  {  pUnk->Release();  return
teInputText;  }

                //.Is it a Input Radio Button?
                theEl->QueryInterface(
IID_IHTMLOptionButtonElement, (LPVOID*)&pUnk );
                if ( pUnk )  {  pUnk->Release();  return
teInputRadio;  }

                // None of the above!
                return teCantTouch;
}


/***********************************************************/
/**  Utility Functions
/***********************************************************/
inline int CTouchCheck::_inside(RECT* theRect, long theX,
long theY)
{
                if ( (theX>=theRect->left) && (theX<=theRect-
>right)  &&
                                  (theY>=theRect->top)   &&
(theY<=theRect->bottom) )
                                return 1;
                else
                                return 0;
}

inline int CTouchCheck::_outside(RECT* theRect, long theX,
long theY)
{
                if ( (theX<theRect->left) || (theX>theRect->right)
||
                                  (theY<theRect->top)   || (theY>theRect-
>bottom) )
                                return 1;
                else
                                return 0;
}

inline void CTouchCheck::_copyRect(RECT * from, RECT * to)
{
                to->left    = from->left;
                to->right   = from->right;
                to->top     = from->top;
                to->bottom = from->bottom;
}

inline void CTouchCheck::_copyRectWithOffset( RECT* from,
RECT* to )
{
                to->left    = from->left    + (p_mScreenToClientX -
p_mScrollLeft);
                to->right   = from->right   + (p_mScreenToClientX -
p_mScrollLeft);
                to->top     = from->top     + (p_mScreenToClientY -
p_mScrollTop );
                to->bottom = from->bottom + (p_mScreenToClientY -
p_mScrollTop );
}


inline void CTouchCheck::_clipRectToClientRect( RECT* r )
{
                if ( r->left   < p_mClientRect.left    )  r->left
= p_mClientRect.left;
                if ( r->right  > p_mClientRect.right   )  r->right
= p_mClientRect.right;
```

```
        if ( r->top     < p_mClientRect.top     )  r->top
= p_mClientRect.top;
        if ( r->bottom > p_mClientRect.bottom )  r->bottom
= p_mClientRect.bottom;
}
```

Feedback.h

```
// FeedBack.h: interface for the CFeedBack class.
//
//////////////////////////////////////////////////////

#ifndef FEEDBACK_H
#define FEEDBACK_H

class CFeedBack : public CObject
{
public:
            CFeedBack();
            virtual ~CFeedBack();
            static CFeedBack *Create();
            void Enable(UINT nID);
            void Disable(UINT nID);

private:
    BOOL StartSerraMouseFake(UINT period);
    void StopSerraMouseFake(void);

private:
    void CFeedBack::BoundsCheck(RECT *r);
            int m_cyRes;
            int m_cxRes;
            BOOL initSuccess;
    HRESULT timerID;
};
#endif // FEEDBACK_H
```

Feedback.cpp

```
// FeedBack.cpp: implementation of the CFeedBack class.
//
///////////////////////////////////////////////////////

#include "stdafx.h"
#include "outdata.h"
#include <mmsystem.h>
#include <assert.h>
#include <winbase.h>
#include "wincomm.h"
#include "FeedBack.h"
#include "TouchCheck.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

#define PERIOD      10
#define SCREEN      65535L

// boolean variables for forces that must be explicitly
turned off
static BOOL f_pushvscroll    = FALSE;
static BOOL f_wmsz           = FALSE;
static BOOL f_dragging       = FALSE;
static BOOL f_hscrolling     = FALSE;
static BOOL f_vscrolling     = FALSE;
static BOOL f_menuitem       = FALSE;

RECT       g_rcObj;
extern CTouchCheck* g_pTouch;

//////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////
CFeedBack::CFeedBack()
{
    TRACE0("CFeedBack::CFeedBack\n");
    timerID = NULL;
    initSuccess = FALSE;

    m_cxRes = GetSystemMetrics(SM_CXSCREEN);
    m_cyRes = GetSystemMetrics(SM_CYSCREEN);

    CommInit();
    // try COM1 thru COM4
    for (int i = 1; i <= 4; i++) {
        if (CommConnect(i,38400l)){
            TRACE1("Connected on COM%d\n", i);
            initSuccess = StartSerraMouseFake(PERIOD);
            break;
        }
    }
}

CFeedBack::~CFeedBack()
{
    TRACE0("CFeedBack::~CFeedBack\n");

    Disable(FBID_OFF);
    StopSerraMouseFake();
    CommEnd();
}

CFeedBack *CFeedBack::Create()
{
    TRACE("CFeedBack::Create\n");

    CFeedBack *m_CFeedback = new CFeedBack();

    if(m_CFeedback->initSuccess){
        return m_CFeedback;
    }else{
        delete m_CFeedback;
        return(NULL);
    }
}

void CFeedBack::BoundsCheck(RECT *r)
{
    if(r->left < 0) r->left = 0;
    if(r->top  < 0) r->top  = 0;
    if(r->right  > m_cxRes) r->right  = m_cxRes;
    if(r->bottom > m_cyRes) r->bottom = m_cyRes;
}

void CFeedBack::Enable(UINT nID)
{
    unsigned char InBuf[10];
    int CommSuccess = 0;
    RECT *r = NULL;

    InBuf[0] = FBID_ON;
    TRACE1("Enable called for FBID #%d:", nID);
    switch (nID) {
        case 305:     // TEXT SELECT
            r = &g_rcObj;
            BoundsCheck(r);
            *(WORD *)(InBuf + 1) = (WORD)(r->left  * SCREEN
/ m_cxRes);
            *(WORD *)(InBuf + 3) = (WORD)(r->top   * SCREEN
/ m_cyRes);
            if(CommSendMsg(nID,InBuf,5)) TRACE(" %d message
sent, (%d,%d)\n",nID,r->left,r->top);
            else                         TRACE1(" ** %d
message failed **\n", nID);
            break;
        case 306:     // PUSH VSCROLL
            if (!f_pushvscroll)
                                 {
                                               r = &g_rcObj;
                                               BoundsCheck(r);
                                               *(WORD *)(InBuf + 1)
= (WORD)(r->left  * SCREEN / m_cyRes);
                                               f_pushvscroll =
CommSendMsg(nID,InBuf,3);
                                               if(f_pushvscroll)
TRACE(" %d message sent, %d\n",nID,r->left);
                                               else
TRACE1(" ** %d message failed **\n", nID);
            }else{
                TRACE0(" FBID_PUSHVSCROLL already
enabled\n");
            }
            break;
        case FBID_WMSZ:
            f_wmsz = CommSendMsg(FBID_WMSZ,InBuf,1);
            if(f_wmsz) TRACE0("FBID_WMSZ message sent\n");
            else       TRACE0("FBID_WMSZ message failed\n");
            break;
#if 0
        case FBID_FOCUS:
            r = &g_rcWnd;
            BoundsCheck(r);
            *(WORD *)(InBuf + 1) = (WORD)( (r->left   + 2) *
SCREEN / m_cxRes);
            *(WORD *)(InBuf + 3) = (WORD)( (r->top    + 2) *
SCREEN / m_cyRes);
            *(WORD *)(InBuf + 5) = (WORD)( (r->right  - 2) *
SCREEN / m_cxRes);
            *(WORD *)(InBuf + 7) = (WORD)( (r->bottom - 2) *
SCREEN / m_cyRes);
            if(CommSendMsg(FBID_FOCUS,InBuf,9))
                TRACE(" FBID_FOCUS message sent
(%d,%d,%d,%d)\n",r->left,r->top,r->right,r->bottom);
            else TRACE0(" FBID_FOCUS message failed\n");
#endif
            break;
        case FBID_DRAGGING:
            if (!f_dragging) {
                f_dragging =
CommSendMsg(FBID_DRAGGING,InBuf,1);
                if(f_dragging) TRACE0(" FBID_DRAGGING
message sent\n");
                else           TRACE0(" FBID_DRAGGING
message failed\n");
```

```
            }else{
                TRACE0(" FBID_DRAGGING already enabled\n");
            }
            break;
        case FBID_HSCROLLING:
            if (!f_hscrolling) {
                r = &g_rcObj;
                BoundsCheck(r);
                *(WORD *)(InBuf + 1) = (WORD)(r->left  *
SCREEN / m_cxRes);
                *(WORD *)(InBuf + 3) = (WORD)(r->Lop   *
SCREEN / m_cyRes);
                *(WORD *)(InBuf + 5) = (WORD)(r->right *
SCREEN / m_cxRes);
                *(WORD *)(InBuf + 7) = (WORD)(r->bottom*
SCREEN / m_cyRes);
                f_hscrolling =
CommSendMsg(FBID_HSCROLLING,InBuf,9);
                if(f_hscrolling) TRACE0("FBID_HSCROLLING
message sent\n");
                else             TRACE0("FBID_HSCROLLING
message failed\n");
            }
            break;
        case FBID_VSCROLLING:
            if (!f_vscrolling) {
                r = &g_rcObj;
                BoundsCheck(r);
                *(WORD *)(InBuf + 1) = (WORD)(r->left  *
SCREEN / m_cxRes);
                *(WORD *)(InBuf + 3) = (WORD)(r->top   *
SCREEN ./ m_cyRes);
                *(WORD *)(InBuf + 5) = (WORD)(r->right *
SCREEN / m_cxRes);
                *(WORD *)(InBuf + 7) = (WORD)(r->bottom*
SCREEN / m_cyRes);
                f_vscrolling =
CommSendMsg(FBID_VSCROLLING,InBuf,9);
                if(f_vscrolling) TRACE0("FBID_VSCROLLING
message sent\n");
                else             TRACE0("FBID_VSCROLLING
message failed\n");
            }
            break;
        case FBID_LISTITEM:   // 2
        case FBID_TREEITEM:   // 3
        case FBID_MENUITEM:   // 6
        case FBID_PUSHBTN:    // 10
        case FBID_CLOSE:      // 402
        case FBID_MAXBTN:     // 407
        case FBID_MINBTN:     // 408
        case FBID_VSCROLL:    // 410
            r = &g_rcObj;
            BoundsCheck(r);
            *(WORD *)(InBuf + 1) = (WORD)(r->left  * SCREEN
/ m_cxRes);
            *(WORD *)(InBuf + 3) = (WORD)(r->top   * SCREEN
/ m_cyRes);
            *(WORD *)(InBuf + 5) = (WORD)(r->right * SCREEN
/ m_cxRes);
            *(WORD *)(InBuf + 7) = (WORD)(r->bottom* SCREEN
/ m_cyRes);
            if(CommSendMsg(nID,InBuf,9)) TRACE(" %d message
sent, (%d,%d,%d,%d)\n",nID,r->left,r->top,r->right,r-
>bottom);
            else                        TRACE1(" ** %d
message failed **\n", nID);
            break;
        case FBID_LISTFOLDER: // 4
        case FBID_ITEMFOCUS:  // 102
        case FBID_TITLEBAR:   // 401
        case FBID_GROWBOX:    // 403
        case FBID_HELP:       // 404
        case FBID_HSCROLL:    // 405
        case FBID_MENU:       // 406
        case FBID_SYSMENU:    // 409
            TRACE1(" %d recognized, but no message sent\n",
nID);
            break;
        default:
            TRACE1(" ** %d unrecognized by
CFeedback::Enable() **\n", nID);
            break;
    }
}

void CFeedBack::Disable(UINT nID)
{
    unsigned char InBuf[10];
    RECT *r = NULL;
    InBuf[0] = FBID_OFF;
    TRACE1("Disable called for FBID #%d:", nID);
    switch (nID) {
        case 305:               // TEXT SELECT
            CommSendMsg(nID,InBuf,1);
            break;
        case 306:               // PUSH VSCROLL
                                if (f_pushvscroll)
                                {
                                        f_pushvscroll =
!CommSendMsg(nID,InBuf,1);
                if(f_pushvscroll) TRACE0(" FBID_PUSHVSCROLL
** FAILED **\n");
                else             TRACE0(" FBID_PUSHVSCROLL
disabled     \n");
            }else{
                TRACE0(" FBID_PUSHVSCROLL not enabled\n");
            }
            break;
        case FBID_WMSZ:
#if 0
            if (f_wmsz) {
                r = &g_rcWnd;
                BoundsCheck(r);
                *(WORD *)(InBuf + 1) = (WORD)( (r->left   +
2) * SCREEN / m_cxRes);
                *(WORD *)(InBuf + 3) = (WORD)( (r->top    +
2) * SCREEN / m_cyRes);
                *(WORD *)(InBuf + 5) = (WORD)( (r->right  -
2) * SCREEN / m_cxRes);
                *(WORD *)(InBuf + 7) = (WORD)( (r->bottom -
2) * SCREEN / m_cyRes);                    f_wmsz =
!(CommSendMsg(FBID_WMSZ,InBuf,9));
                if(f_wmsz)  TRACE0(" FBID_WMSZ ** FAILED
**\n");
                else        TRACE(" FBID_WMSZ disabled
(%d,%d,%d,%d)\n",r->left,r->top,r->right,r->bottom);
            }else{
                TRACE(" FBID_WMSZ not enabled\n");
            }
#endif
            break;
        case FBID_DRAGGING:
            if (f_dragging) {
                f_dragging =
!(CommSendMsg(FBID_DRAGGING,InBuf,1));
                if(f_dragging) TRACE0(" FBID_DRAGGING **
FAILED **\n");
                else           TRACE0(" FBID_DRAGGING
disabled     \n");
            }else{
                TRACE0(" FBID_DRAGGING not enabled\n");
            }
            break;
        case FBID_HSCROLLING:
            if (f_hscrolling) {
                f_hscrolling =
!(CommSendMsg(FBID_HSCROLLING,InBuf,1));
                if(f_hscrolling) TRACE0(" FBID_HSCROLLING **
FAILED **\n");
                else             TRACE0(" FBID_HSCROLLING
disabled     \n");
            }else{
                TRACE0(" FBID_HSCROLLING not enabled\n");
            }
            break;
        case FBID_VSCROLLING:
            if (f_vscrolling) {
                f_vscrolling =
!(CommSendMsg(FBID_VSCROLLING,InBuf,1));
                if(f_vscrolling) TRACE0(" FBID_VSCROLLING **
FAILED **\n");
                else             TRACE0(" FBID_VSCROLLING
disabled     \n");
            }else{
                TRACE0(" FBID_VSCROLLING not enabled\n");
            }
            break;
        case FBID_MENUITEM:
            if(f_menuitem){
                f_menuitem =
!CommSendMsg(FBID_MENUITEM,InBuf,1);
                if(f_menuitem) TRACE(" FBID_MENUITEM **
FAILED **\n");
                else           TRACE(" FBID_MENUITEM
disabled\n");
            }else{
                TRACE(" FBID_MENUITEM not enabled\n");
            }
            break;
        case FBID_OFF:
            if(CommSendMsg(FBID_OFF,InBuf,1)){
                f_vscrolling = f_hscrolling = f_dragging =
FALSE;
                f_wmsz = FALSE;
                TRACE0(" FBID_OFF sent\n");
            }else{
                TRACE0(" FBID_OFF ** failed **\n");
            }
            break;
        default:
            TRACE1(" ** %d unrecognized by
CFeedback::Disable **\n", nID);
            break;
    }
}

void WINAPI TimeFunc(UINT wTimerID, UINT msg, DWORD dwUser,
DWORD dw1, DWORD dw2)
{
    unsigned char InBuf[20];
    WORD dx, dy;
    unsigned char newButtons;
    static unsigned char buttons = 0;
    DWORD flags = 0;
    static WORD scrollDir = 0, scrollCount = 0;
```

```
     UCHAR scrollRate = 0xff;

     POINT cursorPos;

     if(CommGetMsg(InBuf)) { // there's a message
         unsigned short type = *(unsigned short *)(InBuf); //
bytes 0 & 1 are types
         switch(type) {
         case('P'):  // its a position message
             dx = *(WORD *)(InBuf + 2);
             dy = *(WORD *)(InBuf + 4);
             newButtons = InBuf[6];
             if ((newButtons ^ buttons) & 0x1) {      //left
mouse button change
                 if (newButtons & 0x01)  flags |=
MOUSEEVENTF_LEFTDOWN;
                 else                    flags |=
MOUSEEVENTF_LEFTUP;
             }
             if ((newButtons ^ buttons) & 0x2) {      // right
mouse button change
                 if (newButtons & 0x02)  flags |=
MOUSEEVENTF_RIGHTDOWN;
                 else                    flags |=
MOUSEEVENTF_RIGHTUP;
             }
             if ((newButtons ^ buttons) & 0x4) {      //
middle mouse button change
                 if (newButtons & 0x04)  flags |=
MOUSEEVENTF_MIDDLEDOWN;
                 else                    flags |=
MOUSEEVENTF_MIDDLEUP;
             }
             flags |= (MOUSEEVENTF_ABSOLUTE |
MOUSEEVENTF_MOVE);

             mouse_event(flags, dx, dy, 0, 0);
             buttons = newButtons;  //store the last buttons

                              // Do TouchCheck stuff
                              #define _ResetTouchProtected()
           InterlockedExchange(
&(CTouchCheck::p_mTouchProtected), 0 )
                              #define _SetTouchProtected()
           InterlockedExchange(
&(CTouchCheck::p_mTouchProtected), 1 )
                              _SetTouchProtected();
                              if (
CTouchCheck::IsReadyToTouch())
                              {
                                        GetCursorPos(
&cursorPos );
                                        g_pTouch-
>TryTouching( cursorPos.x, cursorPos.y );
                              }
                              _ResetTouchProtected();
             break;
         case('A'):  // an action message
#if 0
             if(GetCursorPos(&cursorPos)){
                 CAccessible *pacc =
CAccessible::Create(cursorPos);
                 if(pacc) pacc->DoObjDefAction();
             }
#endif
             break;
         case('S'):  // a scrolling message
#if 0
                              scrollRate = *(UCHAR *)(InBuf
+ 2);
             scrollDir  = *(UCHAR *)(InBuf + 3);
             TRACE2("Scrolling Message: Rate: %d  Dir: %d\n",
(int)scrollRate, (int)scrollDir);
             scrollCount++;
             if(scrollRate < 200 && scrollCount >
scrollRate){
                 TRACE1("Scrolling Window: Rate: %d\n",
scrollRate);
                 ScrollTheWindow(cursorPos, 0, scrollDir);
                 scrollCount = 0;
             }
#endif
             break;
         } //end switch
     }
}

BOOL CFeedBack::StartSerraMouseFake(UINT period)
{
    TRACE("CFeedBack::StartSerraMouseFake\n");

    TIMECAPS tc;
    if (timeGetDevCaps(&tc, sizeof(TIMECAPS)) !=
TIMERR_NOERROR)     return FALSE;

    UINT wTimerRes = min(max(tc.wPeriodMin,
1),tc.wPeriodMax);
    timeBeginPeriod(wTimerRes);
    if (period < wTimerRes)
        period = wTimerRes;

    timerID = timeSetEvent(period,period,TimeFunc,0,
TIME_PERIODIC);
    if(timerID) return TRUE;
    return FALSE;
}

void CFeedBack::StopSerraMouseFake(void)
{
    TRACE("CFeedBack::StopSerraMouseFake\n");
    if (timerID) timeKillEvent(timerID);
}

---------------------------------------------------------------
```

Vbutil.h

```
// VBUTIL.C - Example DLL for Visual Basic applications.

//@B VBUtil
#include "vbutil.h"
HINSTANCE dllInst = NULL;
// This function is the library entry point. It's
technically
// optional for 32-bit programs, but you'll have more
options later if you define it.

BOOL WINAPI DllMain(HINSTANCE hInstA, DWORD dwReason, LPVOID
lpvReserved)
{
          switch (dwReason) {
          case DLL_PROCESS_ATTACH:
                    // The DLL is being mapped into the
process's address space
                    // Do any additional initialization here
                    dllInst = hInstA;
                    break;

          case DLL_THREAD_ATTACH:
                    // A thread is being created
              break;

          case DLL_THREAD_DETACH:
                    // A thread is exiting cleanly
              break;

          case DLL_PROCESS_DETACH:
                    // The DLL is being unmapped from the
process's address space
                    // Do any additional cleanup here
                    dllInst = 0;
              break;
          }

          return TRUE;
}
//@E VBUtil

// 16-bit version for comparison
#if 0
int PASCAL LibMain(HINSTANCE hInstA, WORD wDataSeg,
                                             WORD cbHeapSize,
LPSTR lpCmdLine)
{
          if (cbHeapSize != 0)
                    UnlockData(0);
          dllInst = hInstA;

          // Do any additional 16-bit server initialization
here
          return dllInst;
}

int FAR PASCAL WEP(int bSystemExit)
{
          // Do any additional 16-bit server cleanup here
          dllInst = 0;
          return 1;
}
#endif

//@B ErrorHandler
void ErrorHandler(Long e)
{
    DWORD err = 0;
    if (e >= 0) {
        err = (DWORD)e;
    } else {
        err = HResultToErr(e);
    }
    SetLastError((DWORD)err);
}
//@E ErrorHandler

DWORD HResultToErr(Long e)
{
    ASSERT(e < 0);

    switch (e) {
    case E_INVALIDARG:
        return ERROR_INVALID_PARAMETER;
    case E_OUTOFMEMORY:
        return ERROR_NOT_ENOUGH_MEMORY;
    case DISP_E_BADINDEX:
        return ERROR_INVALID_INDEX;
    case DISP_E_TYPEMISMATCH:
        return ERROR_INVALID_DATATYPE;
    case DISP_E_EXCEPTION:
        return ERROR_EXCEPTION_IN_SERVICE;
    case DISP_E_BADVARTYPE:
        return ERROR_INVALID_DATATYPE;
    case DISP_E_ARRAYISLOCKED:
        return ERROR_LOCKED;
    case E_UNEXPECTED:
        return ERROR_INVALID_DATA;
    case DISP_E_OVERFLOW:
        return ERROR_ARITHMETIC_OVERFLOW;
    case E_ACCESSDENIED:
        return ERROR_ACCESS_DENIED;
    case E_POINTER:
        return ERROR_INVALID_ADDRESS;
    case E_HANDLE:
        return ERROR_INVALID_HANDLE;
    case E_ABORT:
        return ERROR_OPERATION_ABORTED;
    case E_FAIL:
        return ERROR_GEN_FAILURE;
    }
    return ERROR_INVALID_DATA;
}
```

Vbutil.cpp

```
// VBUTIL.C - Example DLL for Visual Basic applications.

//@B VBUtil
#include "vbutil.h"

HINSTANCE dllInst = NULL;

// This function is the library entry point. It's
technically
// optional for 32-bit programs, but you'll have more
options later
// if you define it.

BOOL WINAPI DllMain(HINSTANCE hInstA, DWORD dwReason, LPVOID
lpvReserved)
{
          switch (dwReason) {
          case DLL_PROCESS_ATTACH:
                    // The DLL is being mapped into the
process's address space
                    // Do any additional initialization here
                    dllInst = hInstA;
                    break;

          case DLL_THREAD_ATTACH:
                    // A thread is being created
              break;

          case DLL_THREAD_DETACH:
                    // A thread is exiting cleanly
              break;

          case DLL_PROCESS_DETACH:
                    // The DLL is being unmapped from the
process's address space
                    // Do any additional cleanup here
                    dllInst = 0;
              break;
          }
          return TRUE;
}
//@E VBUtil

// 16-bit version for comparison
#if 0
int PASCAL LibMain(HINSTANCE hInstA, WORD wDataSeg,
                                           WORD cbHeapSize,
LPSTR lpCmdLine)
{
          if (cbHeapSize != 0)
                    UnlockData(0);
          dllInst = hInstA;

          // Do any additional 16-bit server init here

          return dllInst;
}

int FAR PASCAL WEP(int bSystemExit)
{
          // Do any additional 16-bit server cleanup here
          dllInst = 0;
          return 1;
}
#endif

//@B ErrorHandler
void ErrorHandler(Long e)
{
    DWORD err = 0;
    if (e >= 0) {
        err = (DWORD)e;
    } else {
        err = HResultToErr(e);
    }
    SetLastError((DWORD)err);
}
//@E ErrorHandler

DWORD HResultToErr(Long e)
{
    ASSERT(e < 0);

    switch (e) {
    case E_INVALIDARG:
        return ERROR_INVALID_PARAMETER;
    case E_OUTOFMEMORY:
        return ERROR_NOT_ENOUGH_MEMORY;
    case DISP_E_BADINDEX:
        return ERROR_INVALID_INDEX;
    case DISP_E_TYPEMISMATCH:
        return ERROR_INVALID_DATATYPE;
    case DISP_E_EXCEPTION:
        return ERROR_EXCEPTION_IN_SERVICE;
    case DISP_E_BADVARTYPE:
        return ERROR_INVALID_DATATYPE;
    case DISP_E_ARRAYISLOCKED:
        return ERROR_LOCKED;
    case E_UNEXPECTED:
        return ERROR_INVALID_DATA;
    case DISP_E_OVERFLOW:
        return ERROR_ARITHMETIC_OVERFLOW;
    case E_ACCESSDENIED:
        return ERROR_ACCESS_DENIED;
    case E_POINTER:
        return ERROR_INVALID_ADDRESS;
    case E_HANDLE:
        return ERROR_INVALID_HANDLE;
    case E_ABORT:
        return ERROR_OPERATION_ABORTED;
    case E_FAIL:
        return ERROR_GEN_FAILURE;
    }
    return ERROR_INVALID_DATA;
}
```

OutData.h

```
////////////////////////////////////////////////////////////
// IFData.h - this header file contains all IForce related
data and string definitions

#ifndef   _IFDATA_H_
#define _IFDATA_H_

////////////////////////////////////////////////////////////
// Control Feedback Definitions
////////////////////////////////////////////////////////////
      // currently defined in SerraRemote/Feedback?
#define FBID_OFF                                    0       //  Y
#define FBID_ON                         1

////////////////////////////////////////////////////////////
// Definitions for object related feedback
/////////////////////////////////////////////////////////////
#define   FBID_LISTITEM                             2
#define FBID_TREEITEM                               3
#define FBID_LISTFOLDER                             4
#define FBID_TREEFOLDER                             5
#define FBID_MENUITEM                               6       //  Y
#define FBID_CHECKBTN                                     8
#define FBID_RADIOBTN                                     9
#define   FBID_PUSHBTN                                    10
#define FBID_SEPARATOR                                    11

/////////////////////////////////////////////////////////////
// State related Feedback Definitions
/////////////////////////////////////////////////////////////
#define FBID_ITEMUNAVAILABLE  100
#define FBID_ITEMCHECKED                      101
#define FBID_ITEMFOCUS                                    102
#define FBID_ITEMSELECT                                   103
#define FBID_ITEMCOLLAPSE                     104
#define FBID_ITEMEXPAND                                   105

#define FBID_BOTTOM                                 200
#define FBID_BOTTOMLEFT                             201
#define   FBID_BOTTOMRIGHT            202
#define FBID_LEFT                                   203
#define   FBID_RIGHT                                          204
#define FBID_TOP                                    205
#define FBID_TOPLEFT                                206
#define   FBID_TOPRIGHT                             207
#define FBID_TITLE                                        208

/////////////////////////////////////////////////////////////
// Definitions for event related feedback
/////////////////////////////////////////////////////////////
#define FBID_DRAGGING                               300     //  Y
```

```
#define FBID_WMSZ                           301
#define FBID_HSCROLLING                     302
#define FBID_VSCROLLING                     303
#define FBID_FOCUS                          304
#define FBID_WNDMINMAX                           305
#define FBID_WNDCHANGE                      306        //  Y

/////////////////////////////////////////////////////////////
// Window elements
/////////////////////////////////////////////////////////////
#define FBID_BORDER                400
#define FBID_TITLEBAR              401
#define FBID_CLOSE                 402
#define FBID_GROWBOX               403
#define FBID_HELP                  404
#define FBID_HSCROLL               405
#define FBID_MENU                  406
#define FBID_MAXBTN                407
#define FBID_MINBTN                408
#define FBID_SYSMENU               409
#define FBID_VSCROLL               410

/////////////////////////////////////////////////////////////
// Times
/////////////////////////////////////////////////////////////
typedef struct
{
         int       cOps;
         DWORD dwBgn;
         DWORD dwEnd;
         DWORD dwCum;
} EV_TIME, *PEV_TIME;

typedef struct
{
         int fDrag : 2;
         int fWmsz : 1;
         int fScroll          : 1;
} FLAGS;
/////////////////////////////////////////////////////////////
// Some integer and string ids
/////////////////////////////////////////////////////////////
#define SZ_RECT                                "(%d,%d,%d,%d)"
#define SZ_NUMDRAG                   " :%ld drags"
#define SZ_HORZ                                "Horizontal
scrollbar"
#define SZ_VERT                                "Vertical scrollbar"
#define SZ_SCROLLING                 "%s\nPos: %ld,Min: %ld,Max:
%ld"

#define MAX_BUF                      2048
#define MIN_BUF                      128
#define SMALL                        64

/////////////////////////////////////////////////////////////
// Event strings
/////////////////////////////////////////////////////////////
#define NUM_OP                       "%ld Operations"
#define TIME_OP                      "Operation time: %s\n"
#define TIME_CUM    "Cumulative Time: %s\n\n"
#define TIME_STRING                  "%lu Hr(s):%lu Min(s):%lu
Sec(s): %lu ms"

#define OP_APP                       "\nApp Exited\n"
#define    OP_DRAG                   "\nDrag operation\n"
#define OP_SCROLL    "\nScroll operation\n"
#define OP_WMSZ                      "\nMove/size operation\n"

/////////////////////////////////////////////////////////////
// Event strings
/////////////////////////////////////////////////////////////
#define EV_NONE                                0
#define    EV_LBTNDN                 1
#define EV_LBTNUP                    2
#define    EV_MOUSEMOVE              3
#define EV_MENUSEL                   4
#define EV_DRAGSEL                   5
#define    EV_DRAGGING                         6
#define    EV_DROP                             7
#define EV_SCROLLING                 8
#define EV_WMSZ                                9
#define EV_FOCUS                     10

/////////////////////////////////////////////////////////////
// Object state strings
/////////////////////////////////////////////////////////////
#define STATE_UNAVAILABLE                      "Unavailable;"
#define STATE_SELECTED
          "Selected;"
#define STATE_FOCUSED                                  "Focused;"
#define STATE_PRESSED                                  "Pressed;"
#define STATE_CHECKED                                  "Checked;"
#define STATE_MIXED                                    "Mixed;"
#define STATE_READONLY                                 "Read
only;"
#define STATE_HOTTRACKED                       "Hot tracked;"
#define STATE_DEFAULT                                  "Default;"
#define STATE_EXPANDED
          "Expanded;"
#define STATE_COLLAPSED
          "Collapsed;"
#define STATE_BUSY                                     "Busy;"
#define STATE_FLOATING
          "Floating;"
#define STATE_MARQUEED                                 "Scroll
text;"
#define STATE_ANIMATED
          "Animated;"
#define STATE_INVISIBLE                                "Hidden;"
#define STATE_OFFSCREEN                                "Off-
screen;"
#define STATE_SIZABLE                                  "Sizable;"
#define STATE_MOVABLE                                  "Movable;"
#define STATE_SELFVOICING                    "Self voice;"
#define STATE_SELECTABLE                     "Selectable;"
#define STATE_FOCUSABLE
          "Focusable;"
#define STATE_LINKED                                   "Linked;"
#define STATE_TRAVERSED
          "Tracersed;"
#define STATE_MULTISEL                                 "Multi
sel;"
#define STATE_EXTSEL                                   "Ext sel;"
#define STATE_AL_LO                                    "Low;"
#define STATE_AL_ME                                    "Medium;"
#define STATE_AL_HI                                    "Hi;"

#endif
```

StdAfx.h

```
// stdafx.h : include file for standard system include
files,
//      or project specific include files that are used
frequently, but are changed infrequently
#if
!defined(AFX_STDAFX_H__E9986C44_3FEB_11D1_A868_0060083A2742_
_INCLUDED_)
#define
AFX_STDAFX_H__E9986C44_3FEB_11D1_A868_0060083A2742__INCLUDED
_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define STRICT

#include <afxwin.h>
#include <afxdisp.h>

#define _WIN32_WINNT 0x0400
#define _ATL_APARTMENT_THREADED


#include <atlbase.h>
//You may derive a class from CComModule and use it if you
want to override
//something, but do not change the name of _Module
extern CComModule _Module;
#include <atlcom.h>
#include <atlctl.h>

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_STDAFX_H__E9986C44_3FEB_11D1_A868_0060083A2742_
_INCLUDED)
```

StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard
includes
//  stdafx.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type
information

#include "stdafx.h"

#ifdef _ATL_STATIC_REGISTRY
#include <statreg.h>
#include <statreg.cpp>
#endif

#include <atlimpl.cpp>
#include <atlctl.cpp>
#include <atlwin.cpp>
```

APPENDIX B

Source code implementing different feels of Fig. 12 using force-only ActiveX control.

FeelControl.odl

```
// FeelControl.odl : type library source for ActiveX Control
project.

// This file will be processed by the Make Type Library
(mktyplib) tool to
// produce the type library (FeelControl.tlb) that will
become a resource in FeelControl.ocx.

#include <olectl.h>
#include <idispids.h>

[ uuid(78ACF764-5CC1-11D1-A868-0060083A2742), version(1.0),
  helpfile("FeelControl.hlp"),
  helpstring("FeelControl ActiveX Control module"),
  control ]
library FEELCONTROLLib
{
          importlib(STDOLE_TLB);
          importlib(STDTYPE_TLB);

          //  Primary dispatch interface for
CFeelControlCtrl

          [ uuid(78ACF765-5CC1-11D1-A868-0060083A2742),
            helpstring("Dispatch interface for FeelControl
Control"), hidden ]
          dispinterface _DFeelControl
          {
               properties:
                         // NOTE - ClassWizard will maintain
property information here.
                         //     Use extreme caution when
editing this section.
                         //{{AFX_ODL_PROP(CFeelControlCtrl)
                         [id(1)] BSTR Effect1;
                         [id(2)] BSTR Effect2;
                         [id(3)] BSTR Effect3;
                         [id(4)] BSTR Effect4;
                         [id(5)] BSTR Effect5;
                         [id(6)] BSTR Effect6;
                         //}}AFX_ODL_PROP

               methods:
                         // NOTE - ClassWizard will maintain
method information here.
                         //     Use extreme caution when
editing this section.
               //{{AFX_ODL_METHOD(CFeelControlCtrl)
                         [id(7)] long DoEffect(short
effectNum);
                         [id(8)] long StopEffect(short
effectNum);
                         [id(9)] void StopAll();
                         [id(10)] long SetEffect(short
effectNum, BSTR effectParams);
                         [id(11)] long
DoEnclosureEffect(short effectNum, long left, long top, long
right, long bottom);
                         [id(12)] long ApplyForce(long Xdir,
long Ydir, long Mag );
                         [id(13)] long StopForce();
                         //}}AFX_ODL_METHOD
          };

          //  Event dispatch interface for CFeelControlCtrl
          [ uuid(78ACF766-5CC1-11D1-A868-0060083A2742),
            helpstring("Event interface for FeelControl
Control") ]
          dispinterface _DFeelControlEvents
          {
               properties:
               //  Event interface has no properties

               methods:
               // NOTE - ClassWizard will maintain event
information here.
               //     Use extreme caution when editing this
section.
                         //{{AFX_ODL_EVENT(CFeelControlCtrl)
                         //}}AFX_ODL_EVENT
          };

          //  Class information for CFeelControlCtrl

          [ uuid(5DFDD466-5B37-11D1-A868-0060083A2742),
            helpstring("FeelControl Control"), control ]
          coclass FeelControl
          {
                         [default] dispinterface _DFeelControl;
                         [default, source] dispinterface
_DFeelControlEvents;
          };
          //{{AFX_APPEND_ODL}}
          //}}AFX_APPEND_ODL}}
};
```

FeelControl.def

```
; FeelControl.def : Declares the module parameters.

LIBRARY      "FEELCONTROL.OCX"

EXPORTS
          DllCanUnloadNow      @1 PRIVATE
          DllGetClassObject    @2 PRIVATE
          DllRegisterServer    @3 PRIVATE
          DllUnregisterServer @4 PRIVATE
```

FeelControl.h

```
#if
!defined(AFX_FEELCONTROL_H__78ACF76C_5CC1_11D1_A868_0060083A
2742__INCLUDED_)
#define
AFX_FEELCONTROL_H__78ACF76C_5CC1_11D1_A868_0060083A2742__INC
LUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

// FeelControl.h : main header file for FEELCONTROL.DLL

#if !defined( __AFXCTL_H__ )
          #error include 'afxctl.h' before including this
file
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////
// CFeelControlApp : See FeelControl.cpp for implementation.

class CFeelControlApp : public COleControlModule
{
public:
          BOOL InitInstance();
          int ExitInstance();
};

extern const GUID CDECL _tlid;
extern const WORD _wVerMajor;
extern const WORD _wVerMinor;

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_FEELCONTROL_H__78ACF76C_5CC1_11D1_A868_0060083A
2742__INCLUDED)
```

FeelControl.cpp

```
// FeelControl.cpp : Implementation of CFeelControlApp and
DLL registration.

#include "stdafx.h"
#include "FeelControl.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CFeelControlApp NEAR theApp;

const GUID CDECL BASED_CODE _tlid =
                    { 0x78acf764, 0x5cc1, 0x11d1, { 0xa8,
0x68, 0, 0x60, 0x8, 0x3a, 0x27, 0x42 } };
const WORD _wVerMajor = 1;
const WORD _wVerMinor = 0;

/////////////////////////////////////////////////////////////
// CFeelControlApp::InitInstance - DLL initialization

BOOL CFeelControlApp::InitInstance()
{
          BOOL bInit = COleControlModule::InitInstance();
```

```
        if (bInit)
        {
                // TODO: Add your own module
initialization code here.
        }
        return bInit;
}

/////////////////////////////////////////////////////////////
// CFeelControlApp::ExitInstance - DLL termination

int CFeelControlApp::ExitInstance()
{
        // TODO: Add your own module termination code
here.
        return COleControlModule::ExitInstance();
}

/////////////////////////////////////////////////////////////
// DllRegisterServer - Adds entries to the system registry

STDAPI DllRegisterServer(void)
{
        AFX_MANAGE_STATE(_afxModuleAddrThis);

        if (!AfxOleRegisterTypeLib(AfxGetInstanceHandle(),
_tlid))
                return
ResultFromScode(SELFREG_E_TYPELIB);

        if (!COleObjectFactoryEx::UpdateRegistryAll(TRUE))
                return ResultFromScode(SELFREG_E_CLASS);
        return NOERROR;
}

/////////////////////////////////////////////////////////////
// DllUnregisterServer - Removes entries from the system
registry

STDAPI DllUnregisterServer(void)
{
        AFX_MANAGE_STATE(_afxModuleAddrThis);

        if (!AfxOleUnregisterTypeLib(_tlid, _wVerMajor,
_wVerMinor))
                return
ResultFromScode(SELFREG_E_TYPELIB);

        if
(!COleObjectFactoryEx::UpdateRegistryAll(FALSE))
                return ResultFromScode(SELFREG_E_CLASS);
        return NOERROR;
}
```

FeelForces.h

```
/************************************************************
 * FeelControl
 * (c) 1997 Immersion Corporation
 *
 * FILE
 *              FeelForces.h
 * DESCRIPTION
 *  Provide methods for doing force-feedback with the
ForceClasses
*/

#ifndef    __FEELFORCES_H
#define __FEELFORCES_H

BOOL FeelSetup( HINSTANCE hInst, HWND hWnd );
BOOL FeelCleanup( void );

BOOL FeelBeginEffect( short effectNum );
BOOL FeelEndEffect( short effectNum );
void FeelEndAllEffects( void );

long FeelBeginForce( long Xdir, long Ydir, long Mag );
long FeelEndForce( void );

long FeelEnclosureEffect(short effectNum, long left, long
top, long right, long bottom);

#endif    __FEELFORCES_H
```

FeelForces.cpp

```
/************************************************************
 * FeelControl
 * (c) 1997-1998 Immersion Corporation
 *
 * FILE
 *              FeelForces.cpp
 * DESCRIPTION
 *      Provide methods for doing force-feedback with the
ForceClasses, giving the FeelControl some guts...
*/

#include "stdafx.h"
#include "FeelForces.h"
#include "ForceFeelitMouse.h"
#include "ForceEffect.h"
#include "ForcePeriodic.h"
#include "ForceDamper.h"
#include "ForceEllipse.h"
#include "ForceCondition.h"
#include "ForceConstant.h"
#include "ForceTexture.h"
#include "ForceEnclosure.h"
#include "ForceSpring.h"
#include <stdio.h>

// GLOBAL VARIABLES
CForceFeelitMouse*  gMouse     = NULL;
CForceConstant*     gForce                  = NULL;
CForceEffect*       gEffect1   = NULL;
CForceEffect*       gEffect2   = NULL;
CForceEffect*       gEffect3   = NULL;
CForceEffect*       gEffect4   = NULL;
CForceEffect*       gEffect5   = NULL;
CForceEffect*       gEffect6   = NULL;
CForceEffect*       gEffect7   = NULL;
CForceEffect*       gEffect8   = NULL;
CForceEffect*       gEffect9   = NULL;
CForceEffect*       gEffect11  = NULL;
CForceEffect*       gEngineEnc              = NULL;
FORCE_ENVELOPE      fEnvelope1;
FORCE_ENVELOPE      fEnvelope2;
FORCE_ENVELOPE      fEnvelope3;
FORCE_ENVELOPE      fEnvelope4;


/*
 *  Globals for our params  .
 */
// Laser
DWORD LDIRX  = 1,  LDIRY  = 1, LDUR  = 1000, LMAG  = 10000,
LPER  = 158, LOFF  = 0, LPHA  = 0;
DWORD LDIRX2 = -1, LDIRY2 = 1, LDUR2 = 1000, LMAG2 = 6744,
LPER2 = 13,  LOFF2 = 0, LPHA2 = 0;
// ICE
DWORD IVIS = -4000, ISAT = 8000, IVEL = 10;
// METEOR
DWORD MSTIFF =4000, MWW = 20, MSAT = 8000;
// DENIM TEXTURE
DWORD DPOSK = 8000, DNEGK = 8000, DPOSS = 9, DNEGS = 9,
DDEAD = 3;
// DENIM GRID
DWORD DGPOSK=3000, DGNEGK=3000, DGPOSS=3000, DGNEGS=3000,
DGDEAD=14;
// ENGINE
DWORD EDIRX = 1,  EDIRY = 0,  EDUR=2000,  EMAG=5968,
EPER=295,  EOFF=0,  EPHA=0;
DWORD EDIRX2 = 0, EDIRY2 = 1, EDUR2=3500, EMAG2=10000,
EPER2=100, EOFF2=0, EPHA2=00;
// ENGINE ENCLOSURE
DWORD ESTIFF =9800, EWW = 28, ESAT = 9800; // 8000,20,8000
// RAQUET STRING GRID
DWORD SPOSK=3000, SNEGK=3000, SPOSS=3000, SNEGS=3000,
SDEAD=14;
// RAQUET ELLIPSE
DWORD RSTIFF = 6000, RWW = 10, RSAT = 8000; // RSTIFF =
6000, RWW = 20, RSAT = 8000
// ENGINE ENVELOPE
DWORD EEAL  = 10000, EEAT  = 390697, EEFL =1,  EEFT=967441;
DWORD EEAL2 = 0, EEAT2 = 2572093, EEFL2=10000, EEFT2=830232;
// LASER ENVELOPE
DWORD LEAL   = 3953, LEAT   = 144186, LEFL=387,
LEFT=641860;
DWORD LEAL2  = 10000, LEAT2  = 283720, LEFL2=0,  LEFT2=0;


BOOL FeelSetup( HINSTANCE hInst, HWND hWnd )
{
        BOOL success;
        // Try to get parameters from effects.dat
/*
        FILE *fp = fopen("feelcontrol.dat", "r");
        if (fp) {
                // Laser
                fscanf(fp,"%d %d %d %d %d %d %d",
&LDIRX,  &LDIRY,  &LDUR,  &LMAG,  &LPER,  &LOFF,  &LPHA );
                fscanf(fp,"%d %d %d %d %d %d %d",
&LDIRX2, &LDIRY2, &LDUR2, &LMAG2, &LPER2, &LOFF2, &LPHA2 );
                // ICE
                fscanf(fp,"%d %d %d", &IVIS, &ISAT,
&IVEL );
                // METEOR
                fscanf(fp,"%d %d %d", &MSTIFF, &MWW,
&MSAT );
                // DENIM
                fscanf(fp,"%d %d %d %d %d", &DPOSK,
&DNEGK, &DPOSS, &DNEGS, &DDEAD );
                // ENGINE
                fscanf(fp,"%d %d %d %d %d %d %d",
&EDIRX,  &EDIRY,  &EDUR,  &EMAG,  &EPER,  &EOFF,  &EPHA  );
                // RAQUET STRING GRID
```

```
                     fscanf(fp,"%d %d %d %d %d", &SPOSK,
&SNEGK, &SPOSS, &SNEGS, &SDEAD );
                     // RAQUET ELLIPSE
                     fscanf(fp,"%d %d %d", &RSTIFF, &RWW,
&RSAT );
                     // ENGINE 2
                     fscanf(fp,"%d %d %d %d %d %d %d",
&EDIRX2, &EDIRY2, &EDUR2, &EMAG2, &EPER2, &EOFF2, &EPHA2 );
                     // ENGINE ENVELOPE
                     fscanf(fp,"%d %d %d %d", &EEAL,  &EEAT,
&EEFL,   &EEFT );
                     fscanf(fp,"%d %d %d %d", &EEAL2, &EEAT2,
&EEFL2,  &EEFT2 );
                     // LASER ENVELOPE
                     fscanf(fp,"%d %d %d %d", &LEAL,  &LEAT,
&LEFL,   &LEFT  );
                     fscanf(fp,"%d %d %d %d", &LEAL2, &LEAT2,
&LEFL2,  &LEFT2 );
                     // Close it...
                     fclose(fp);
          } */

          // Set up the Mouse
          gMouse = new CForceFeelitMouse();
          if ( ! gMouse ) goto FS_Err;
          success = gMouse->Initialize( hInst, hWnd );
          if ( ! success ) goto FS_Err;

          // Set up the Force
//        gForce = new CForceConstant();
//        if ( ! gForce ) goto FS_Err;
//        success = gForce->Initialize(
//                   gMouse,
//                   FORCE_CONSTANT_DEFAULT_DIRECTION,
//                   INFINITE,
//                   0
//  );
//        if ( ! success ) goto FS_Err;

// Set envelopes...
          // Envelope 1
          fEnvelope1.dwSize                        =
sizeof(FORCE_ENVELOPE);
          fEnvelope1.dwAttackLevel                 = EEAL;
          fEnvelope1.dwAttackTime                  = EEAT;
          fEnvelope1.dwFadeLevel                   = EEFL;
          fEnvelope1.dwFadeTime                    = EEFT;
          // Envelope 2
          fEnvelope2.dwSize                        =
sizeof(FORCE_ENVELOPE);
          fEnvelope2.dwAttackLevel                 = EEAL2;
          fEnvelope2.dwAttackTime                  = EEAT2;
          fEnvelope2.dwFadeLevel                   = EEFL2;
          fEnvelope2.dwFadeTime                    = EEFT2;
          // Envelope 3
          fEnvelope3.dwSize                        =
sizeof(FORCE_ENVELOPE);
          fEnvelope3.dwAttackLevel                 = LEAL;
          fEnvelope3.dwAttackTime                  = LEAT;
          fEnvelope3.dwFadeLevel                   = LEFL;
          fEnvelope3.dwFadeTime                    = LEFT;
          // Envelope 4
          fEnvelope4.dwSize                        =
sizeof(FORCE_ENVELOPE);
          fEnvelope4.dwAttackLevel                 = LEAL2;
          fEnvelope4.dwAttackTime                  = LEAT2;
          fEnvelope4.dwFadeLevel                   = LEFL2;
          fEnvelope4.dwFadeTime                    = LEFT2;

// Create effect 1 = LASER (PERIODIC SINE {1,0} 750 3023 10
0 0)
          //Laser Effect #1
          gEffect1 = new CForcePeriodic(GUID_Force_Square);
          if ( ! gEffect1 ) goto FS_Err;
          success = ((CForcePeriodic*)gEffect1)->Initialize(
                     gMouse,
                     LMAG,
          // = FORCE_PERIODIC_DEFAULT_MAGNITUDE
                     LPER,
          // = FORCE_PERIODIC_DEFAULT_PERIOD
                     LDUR,
          // = FORCE_PERIODIC_DEFAULT_DURATION
                     LDIRX,
          // X Direction
                     LDIRY,
          // Y Direction
                     LOFF,
          // = FORCE_PERIODIC_DEFAULT_OFFSET
                     LPHA,
          // = FORCE_PERIODIC_DEFAULT_PHASE
                     &fEnvelope3
          );
          if ( ! success ) goto FS_Err;
// Laser effect #2
          gEffect8 = new CForcePeriodic(GUID_Force_Sine);
          if ( ! gEffect8 ) goto FS_Err;
          success = ((CForcePeriodic*)gEffect8)->Initialize(
                     gMouse,
                     LMAG2,
          // = FORCE_PERIODIC_DEFAULT_MAGNITUDE
                     LPER2,
          // = FORCE_PERIODIC_DEFAULT_PERIOD
                     LDUR2,
          // = FORCE_PERIODIC_DEFAULT_DURATION
                     LDIRX2,
          // X Direction
                     LDIRY2,
          // Y Direction
                     LOFF2,
          // = FORCE_PERIODIC_DEFAULT_OFFSET
                     LPHA2,
          // = FORCE_PERIODIC_DEFAULT_PHASE
                     &fEnvelope4
          );
          if ( ! success ) goto FS_Err;

// Create effect 2 = ICE (DAMPER -1000 8000 0 -1)
          gEffect2 = new CForceDamper();
          if ( ! gEffect2 ) goto FS_Err;
          success = ((CForceDamper*)gEffect2)->Initialize(
                     gMouse,
                     IVIS, // =
FORCE_DAMPER_DEFAULT_VISCOSITY
                     ISAT, // =
FORCE_DAMPER_DEFAULT_SATURATION
                     IVEL, // =
FORCE_DAMPER_DEFAULT_MIN_VELOCITY
                     FORCE_EFFECT_AXIS_BOTH
          );
          if ( ! success ) goto FS_Err;

// Create effect 4 = METEOR (ELLIPSE -1 -1 2000 -1 -1 -1 -1
-1 8)
          gEffect4 = new CForceEllipse;
          if ( ! gEffect4 ) goto FS_Err;
          success = ((CForceEllipse*)gEffect4)->Initialize(
                     gMouse,
                     FORCE_ELLIPSE_DEFAULT_WIDTH,
                     FORCE_ELLIPSE_DEFAULT_HEIGHT,
                     MSTIFF, // =
FORCE_ELLIPSE_DEFAULT_STIFFNESS
                     MWW,
                     MSAT,
                     FEELIT_FSTIFF_OUTBOUNDANYWALL,
          //FORCE_ELLIPSE_DEFAULT_STIFFNESS_MASK,
                     FORCE_ELLIPSE_DEFAULT_CLIPPING_MASK,
                     FORCE_ELLIPSE_DEFAULT_CENTER_POINT,
                     NULL
          );
          if ( ! success ) goto FS_Err;

// Create effect 5 = DENIM (CONDITION TEXTURE ???? )
          gEffect5 = new CForceTexture();
          if ( ! gEffect5 ) goto FS_Err;
          success = ((CForceTexture*)gEffect5)->InitTexture(
                     gMouse,
                     DPOSK,
          // lPosBumpMag = FORCE_TEXTURE_DEFAULT_MAGNITUDE,
        DPOSS,
          // dwPosBumpWidth = FORCE_TEXTURE_DEFAULT_WIDTH,
        DDEAD,
          // dwPosBumpSpacing =
FORCE_TEXTURE_DEFAULT_SPACING,
        DNEGK,
          // lNegBumpMag = FORCE_TEXTURE_DEFAULT_MAGNITUDE,
        DNEGS,
          // dwNegBumpWidth = FORCE_TEXTURE_DEFAULT_WIDTH,
        DDEAD,
          // dwNegBumpSpacing=FORCE_TEXTURE_DEFAULT_SPACING,
          FORCE_EFFECT_AXIS_X              // dwfAxis =
FORCE_EFFECT_AXIS_BOTH,
          // pntOffset = FORCE_TEXTURE_DEFAULT_OFFSET_POINT,
          // lDirectionX = FORCE_EFFECT_DEFAULT_DIRECTION_X,
          // lDirectionY = FORCE_EFFECT_DEFAULT_DIRECTION_Y
          );
          if ( ! success ) goto FS_Err;

// Create effect 11 = DENIM (GRID)
          gEffect11 = new CForceCondition( GUID_Force_Grid
);
          if ( ! gEffect11 ) goto FS_Err;
          success = ((CForceCondition*)gEffect11)-
>InitCondition(
                     gMouse,
                     DGPOSK,    //PosK
                     DGNEGK,    //NegK
                     DGPOSS,    //PosSat
                     DGNEGS,    //NegSat
                     DGDEAD,    //Deadband - grid spacing in
pixels
                     FORCE_EFFECT_AXIS_X
//FORCE_EFFECT_AXIS_BOTH
                     //FORCE_CONDITION_DEFAULT_CENTER_POINT
          );
          if ( ! success ) goto FS_Err;

// Create effect 6 = MOTOR (PERIODIC SQUARE {1, 1} 10000
6500 20 0 180)
          gEffect6 = new CForcePeriodic(GUID_Force_Square);
          if ( ! gEffect6 ) goto FS_Err;
          success = ((CForcePeriodic*)gEffect6)->Initialize(
                     gMouse,
                     EMAG,
          // = FORCE_PERIODIC_DEFAULT_MAGNITUDE
```

```
                    EPER,
            // = FORCE_PERIODIC_DEFAULT_PERIOD
                    EDUR,
            // = FORCE_PERIODIC_DEFAULT_DURATION
                    EDIRX,
            // X Direction
                    EDIRY,
            // Y Direction
                    EOFF,
            // = FORCE_PERIODIC_DEFAULT_OFFSET
                    EPHA,
            // = FORCE_PERIODIC_DEFAULT_PHASE
                    &fEnvelope1
            );
            if ( ! success ) goto FS_Err;

    // Create effect 9 = MOTOR (PERIODIC SQUARE (1, 1) 10000
    6500 20 0 180)
            gEffect9 = new CForcePeriodic(GUID_Force_Sine);
            if ( ! gEffect9 ) goto FS_Err;
            success = ((CForcePeriodic*)gEffect9)->Initialize(
                    gMouse,
                    EMAG2,
            // = FORCE_PERIODIC_DEFAULT_MAGNITUDE
                    EPER2,
            // = FORCE_PERIODIC_DEFAULT_PERIOD
                    EDUR2,
            // = FORCE_PERIODIC_DEFAULT_DURATION
                    EDIRX2,
            // X Direction
                    EDIRY2,
            // Y Direction
                    EOFF2,
            // = FORCE_PERIODIC_DEFAULT_OFFSET
                    EPHA2,
            // = FORCE_PERIODIC_DEFAULT_PHASE
                    &fEnvelope2
            );
            if ( ! success ) goto FS_Err;

    // Create effect 7 = RACQUET_GRID (CONDITION GRID)
            gEffect7 = new CForceCondition( GUID_Force_Grid );
            if ( ! gEffect7 ) goto FS_Err;
            success = ((CForceCondition*)gEffect7)-
    >InitCondition(
                    gMouse,
                    SPOSK,      //PosK
                    SNEGK,      //NegK
                    SPOSS,      //PosSat
                    SNEGS,      //NegSat
                    SDEAD //Deadband -grid spacing in pixels
                    //FORCE_EFFECT_AXIS_BOTH,
                    //FORCE_CONDITION_DEFAULT_CENTER_POINT
            );
            if ( ! success ) goto FS_Err;

            // Create effect 3 = RACQUET (ELLIPSE -1 -1 2000 -
    1 -1 -1 -1 -1 8)
            // Make 3 after 7 because 3 is dependent on 7
            gEffect3 = new CForceEllipse;
            if ( ! gEffect3 ) goto FS_Err;
            success = ((CForceEllipse*)gEffect3)->Initialize(
                    gMouse,
                    FORCE_ELLIPSE_DEFAULT_WIDTH,
                    FORCE_ELLIPSE_DEFAULT_HEIGHT,
                    RSTIFF, // =
    FORCE_ELLIPSE_DEFAULT_STIFFNESS
                    RWW,
                    RSAT,
                    FEELIT_FSTIFF_OUTBOUNDANYWALL,
    //FEELIT_FSTIFF_ANYWALL,
            //FORCE_ELLIPSE_DEFAULT_STIFFNESS_MASK,
                    FORCE_ELLIPSE_DEFAULT_CLIPPING_MASK,
                    FORCE_ELLIPSE_DEFAULT_CENTER_POINT,
                    gEffect7
            );
            if ( ! success ) goto FS_Err;

            // Engine Enclosure
            gEngineEnc = new CForceEnclosure;
            if ( ! gEngineEnc ) goto FS_Err;
            success = ((CForceEnclosure*)gEngineEnc)-
    >Initialize(
                    gMouse,
                    FORCE_ENCLOSURE_DEFAULT_WIDTH,
                    FORCE_ENCLOSURE_DEFAULT_HEIGHT,
                    ESTIFF, // =
    FORCE_ELLIPSE_DEFAULT_STIFFNESS
                    ESTIFF,
                    EWW,
                    EWW,
                    ESAT,
                    ESAT,
                    FEELIT_FSTIFF_OUTBOUNDANYWALL,
            //FORCE_ELLIPSE_DEFAULT_STIFFNESS_MASK,
                    FORCE_ENCLOSURE_DEFAULT_CLIPPING_MASK,
                    FORCE_ENCLOSURE_DEFAULT_CENTER_POINT,
                    NULL
            );
            if ( ! success ) goto FS_Err;

            // We're okay!
            return TRUE;
```

```
FS_Err:
            // There were some problems... let's cleanup and
declare ourselves dead!
            FeelCleanup();
            return FALSE;
}

BOOL FeelCleanup( void )
{
            if ( gEngineEnc){ gEngineEnc->Stop();delete
gEngineEnc;         gEngineEnc=NULL; }
            if ( gForce )       { gForce->Stop();    delete
gForce;             gForce      = NULL; }
            if ( gEffect1 )     { gEffect1->Stop();  delete
gEffect1; gEffect1 = NULL; }
            if ( gEffect2 )     { gEffect2->Stop();  delete
gEffect2; gEffect2 = NULL; }
            if ( gEffect3 )     { gEffect3->Stop();  delete
gEffect3; gEffect3 = NULL; }
            if ( gEffect4 )     { gEffect4->Stop();  delete
gEffect4; gEffect4 = NULL; }
            if ( gEffect5 )     { gEffect5->Stop();  delete
gEffect5; gEffect5 = NULL; }
            if ( gEffect6 )     { gEffect6->Stop();  delete
gEffect6; gEffect6 = NULL; }
            if ( gEffect7 )     { gEffect7->Stop();  delete
gEffect7; gEffect7 = NULL; }
            if ( gEffect8 )     { gEffect8->Stop();  delete
gEffect8; gEffect8 = NULL; }
            if ( gEffect9 )     { gEffect9->Stop();  delete
gEffect9; gEffect9 = NULL; }
            if ( gEffect11)     { gEffect11->Stop(); delete
gEffect11;          gEffect11 = NULL; }
            if ( gMouse )       {
                    delete gMouse;              gMouse   =
NULL; }
            return TRUE;
}


BOOL FeelBeginEffect( short effectNum )
{
     switch ( effectNum )
     {
            case 1:
                    if ( gEffect1 )
                        return gEffect1->Start();
                    break;
            case 2:
                    if ( gEffect2 )
                        return gEffect2->Start();
                    break;
            case 3:
                    if ( gEffect3 )
                        return gEffect3->Start();
                    break;
            case 4:
                    if ( gEffect4 )
                        return gEffect4->Start();
                    break;
            case 5:
                    if ( gEffect5 )
                        return gEffect5->Start();
                    break;
            case 6:
                    if ( gEffect6 )
                        return gEffect6->Start();
                    break;
            case 7:
                    if ( gEffect7 )
                        return gEffect7->Start();
                    break;
            case 8:
                    if ( gEffect8 )
                        return gEffect8->Start();
                    break;
            case 9:
                    if ( gEffect9 )
                        return gEffect9->Start();
                    break;
            case 11:
                    if ( gEffect11 )
                        return gEffect11->Start();
                    break;
            default:
                    break;
     }
     return FALSE;
}


BOOL FeelEndEffect( short effectNum )
{
     switch ( effectNum )
     {
            case 1:
                    if ( gEffect1 )
                        return gEffect1->Stop();
                    break;
            case 2:
                    if ( gEffect2 )
                        return gEffect2->Stop();
                    break;
            case 3:
                    if ( gEffect3 )
                        return gEffect3->Stop();
                    break;
            case 4:
                    if ( gEffect4 )
                        return gEffect4->Stop();
                    break;
            case 5:
                    if ( gEffect5 )
                        return gEffect5->Stop();
                    break;
            case 6:
                    if ( gEffect6 )
                        return gEffect6->Stop();
                    break;
            case 7:
                    if ( gEffect7 )
                        return gEffect7->Stop();
                    break;
            case 8:
                    if ( gEffect8 )
                        return gEffect8->Stop();
                    break;
            case 9:
                    if ( gEffect9 )
                        return gEffect9->Stop();
                    break;
            case 10:
                    if ( gEngineEnc )
                        return gEngineEnc->Stop();
                    break;
            case 11:
                    if ( gEffect11 )
                        return gEffect11->Stop();
                    break;
            default:
                    break;
        }
        return FALSE;
}

void FeelEndAllEffects( void )
{
        if ( gEffect1 ) gEffect1->Stop();
        if ( gEffect2 ) gEffect2->Stop();
        if ( gEffect3 ) gEffect3->Stop();
        if ( gEffect4 ) gEffect4->Stop();
        if ( gEffect5 ) gEffect5->Stop();
        if ( gEffect6 ) gEffect6->Stop();
        if ( gEffect7 ) gEffect7->Stop();
        if ( gEffect8 ) gEffect8->Stop();
        if ( gEffect9 ) gEffect9->Stop();
        if ( gEffect11) gEffect11->Stop();
        if ( gForce )       gForce->Stop();
        if ( gEngineEnc)gEngineEnc->Stop();
}

long FeelEnclosureEffect(short effectNum, long left, long
top, long right, long bottom)
{
        // Prepare a rect...
        RECT r = { left, top, right, bottom };

        // Try doing the enclosure, depending on which
effect they want...
        switch ( effectNum )
        {
                BOOL success;
                case 3:
                    if ( ! gEffect3 ) return FALSE;
                    success =
((CForceEllipse*)gEffect3)->ChangeParameters(
        &r,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        (CForceEffect*) FORCE_EFFECT_DONT_CHANGE
        );
                    if ( ! success ) return FALSE;
                    return gEffect3->Start();
                    break;
                case 4:
                    if ( ! gEffect4 ) return FALSE;
                    success =
((CForceEllipse*)gEffect4)->ChangeParameters(
        &r,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        (CForceEffect*) FORCE_EFFECT_DONT_CHANGE
        );
                    if ( ! success ) return FALSE;
                    return gEffect4->Start();
                    break;
                case 10:
                    if ( ! gEngineEnc ) return FALSE;
                    success =
((CForceEnclosure*)gEngineEnc)->ChangeParameters(
        &r,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        FORCE_EFFECT_DONT_CHANGE,
        (CForceEffect*) FORCE_EFFECT_DONT_CHANGE
        );
                    if ( ! success ) return FALSE;
                    return gEngineEnc->Start();
                    break;
                // These effects don't use enclosures!
                case 1:
                case 2:
                case 5:
                case 6:
                case 7:
                case 8:
                case 9:
                default:
                    return FALSE;
        }
        return FALSE;       // Have this just in case...
}


long FeelBeginForce( long Xdir, long Ydir, long Mag )
{
//      if ( gForce )
//      {
//              POINT myPt = { Xdir, Ydir };
//              gForce->ChangeParameters(
//                  myPt,
//                  FORCE_EFFECT_DONT_CHANGE,
//                  Mag
//              );
//      }
        return 0;
}

long FeelEndForce( void )
{
//      if ( gForce )
//              return gForce->Stop();
        return 0;
}
```

----

howtodat.txt

```
This file shows an outline of the effects.dat file.
Everything is a DWORD and describes a parameter for
the initialization of an effect.

Here's the order:
// LASER Periodic
// ICE Damper
// METEOR Ellipse
// DENIM Texture
// ENGINE Periodic
// RAQUET STRING Grid
// RAQUET Ellipse (uses Grid)


Here's the actual value descriptions (in shorthand):
LDUR      LMAG      LPER      LOFF      LPHA
IVIS      ISAT      IVEL
MSTIFF    MWW                 MSAT
DPOSK     DNEGK     DPOSS     DNEGS     DDEAD
EDIRX     EDIRY     EDUR      EMAG      EPER      EOFF
          EPHA
SPOSK     SNEGK     SPOSS     SNEGS     SDEAD
RSTIFF    RWW                 RSAT


Here's the default values:
// Laser
DWORD LDUR = 400, LMAG = 4000, LPER = 10, LOFF = 0, LPHA =0;
// ICE
DWORD IVIS = -4000, ISAT = 8000, IVEL = 10;
// METEOR
DWORD MSTIFF =4000, MWW = 20, MSAT = 8000;
// DENIM
DWORD DPOSK = 8000, DNEGK = 8000, DPOSS = 50, DNEGS = 50,
DDEAD = 11;
// ENGINE
DWORD EDIRX = 1, EDIRY = 1, EDUR=10000, EMAG=4500,
EPER=20,EOFF=0, EPHA=180;
// RAQUET STRING GRID
DWORD SPOSK=3000, SNEGK=3000, SPOSS=3000, SNEGS=3000,
SDEAD=20;
// RAQUET ELLIPSE
DWORD RSTIFF = 2000, RWW = 20, RSAT = 8000;
-


Here's the initialization code that uses them:
```

```
// Create effect 1 = LASER (PERIODIC SINE (1,0) 750 3023 10
0 0)
        gEffect1 = new CForcePeriodic();
        if ( ! gEffect1 ) goto FS_Err;
        success = ((CForcePeriodic*)gEffect1)->Initialize(
                gMouse,
                GUID_Sine,
                FORCE_PERIODIC_DEFAULT_DIRECTION,
                LDUR,       // =
FORCE_PERIODIC_DEFAULT_DURATION
                LMAG,       // =
FORCE_PERIODIC_DEFAULT_MAGNITUDE
                LPER,       // =
FORCE_PERIODIC_DEFAULT_PERIOD
                LOFF,       // =
FORCE_PERIODIC_DEFAULT_OFFSET
                LPHA        // =
FORCE_PERIODIC_DEFAULT_PHASE
    );
        if ( ! success ) goto FS_Err;

// Create effect 2 = ICE (DAMPER -1000 8000 0 -1)
        gEffect2 = new CForceDamper();
        if ( ! gEffect2 ) goto FS_Err;
        success = ((CForceDamper*)gEffect2)->Initialize(
                gMouse,
                IVIS, // =
FORCE_DAMPER_DEFAULT_VISCOSITY
                ISAT, // =
FORCE_DAMPER_DEFAULT_SATURATION
                IVEL, // =
FORCE_DAMPER_DEFAULT_MIN_VELOCITY
                FORCE_EFFECT_AXIS_BOTH
        );
        if ( ! success ) goto FS_Err;

// Create effect 4 = METEOR (ELLIPSE -1 -1 2000 -1 -1 -1 -1
-1 8)
        gEffect4 = new CForceEllipse;
        if ( ! gEffect4 ) goto FS_Err;
        success = ((CForceEllipse*)gEffect4)->Initialize(
                gMouse,
                FORCE_ELLIPSE_DEFAULT_WIDTH,
                FORCE_ELLIPSE_DEFAULT_HEIGHT,
                MSTIFF, // =
FORCE_ELLIPSE_DEFAULT_STIFFNESS
                MWW,
                MSAT,
                SERRA_FSTIFF_OUTBOUNDANYWALL,
        //FORCE_ELLIPSE_DEFAULT_STIFFNESS_MASK,
                FORCE_ELLIPSE_DEFAULT_CLIPPING_MASK,
                FORCE_ELLIPSE_DEFAULT_CENTER_POINT,
                NULL
        );
        if ( ! success ) goto FS_Err;

// Create effect 5 = DENIM (CONDITION TEXTURE ???? )
        gEffect5 = new CForceCondition;
        if ( ! gEffect5 ) goto FS_Err;
        success = ((CForceCondition*)gEffect5)-
>Initialize(
                gMouse,
                GUID_Serra_Texture,
                DPOSK,      //PosK
                DNEGK,      //NegK
                DPOSS,      //PosSat - period in pixels
                DNEGS,      //NegSat - period in pixels
                DDEAD,      //Deadband - no bump in pixels
                FORCE_EFFECT_AXIS_X,
                FORCE_CONDITION_DEFAULT_CENTER_POINT
        );
        if ( ! success ) goto FS_Err;

// Create effect 6 = MOTOR (PERIODIC SQUARE (1, 1) 10000
6500 20 0 180)
        gEffect6 = new CForcePeriodic();
        if ( ! gEffect6 ) goto FS_Err;
        success = ((CForcePeriodic*)gEffect6)->Initialize(
                gMouse,
                GUID_Square,
                CPoint(EDIRX,EDIRY),// =
FORCE_PERIODIC_DEFAULT_DIRECTION
                EDUR,               // =
FORCE_PERIODIC_DEFAULT_DURATION
                EMAG,               // =
FORCE_PERIODIC_DEFAULT_MAGNITUDE
                EPER,               // =
FORCE_PERIODIC_DEFAULT_PERIOD
                EOFF,               // =
FORCE_PERIODIC_DEFAULT_OFFSET
                EPHA                // =
FORCE_PERIODIC_DEFAULT_PHASE
        );
        if ( ! success ) goto FS_Err;

// Create effect 7 = RACQUET_GRID (CONDITION GRID)
        gEffect7 = new CForceCondition;
        if ( ! gEffect7 ) goto FS_Err;
        success = ((CForceCondition*)gEffect7)-
>Initialize(
                gMouse,
                GUID_Serra_Grid,
                SPOSK,      //PosK
                SNEGK,      //NegK
                SPOSS,      //PosSat
                SNEGS,      //NegSat
                SDEAD,      //Deadband - grid spacing in
pixels
                FORCE_EFFECT_AXIS_BOTH,
                FORCE_CONDITION_DEFAULT_CENTER_POINT
        );
        if ( ! success ) goto FS_Err;

        // Create effect 3 = RACQUET (ELLIPSE -1 -1 2000 -
1 -1 -1 -1 -1 8)
        // Make 3 after 7 because 3 is dependent on 7
        gEffect3 = new CForceEllipse;
        if ( ! gEffect3 ) goto FS_Err;
        success = ((CForceEllipse*)gEffect3)->Initialize(
                gMouse,
                FORCE_ELLIPSE_DEFAULT_WIDTH,
                FORCE_ELLIPSE_DEFAULT_HEIGHT,
                RSTIFF, // =
FORCE_ELLIPSE_DEFAULT_STIFFNESS
                RWW,
                RSAT,
                SERRA_FSTIFF_OUTBOUNDANYWALL,
        //FORCE_ELLIPSE_DEFAULT_STIFFNESS_MASK,
                FORCE_ELLIPSE_DEFAULT_CLIPPING_MASK,
                FORCE_ELLIPSE_DEFAULT_CENTER_POINT,
                gEffect7
        );
        if ( ! success ) goto FS_Err;
```

---

effects.dat

```
1          1          1000       10000      158        0
           0
-1         1          1000       6744       13         0
           0
-4000      8000       10
4000       20         8000
8000       8000       50         50          11
0          1          2000       5968 295    0         0
3000       3000       3000       3000 20
2000       20         8000
1          0          3500       10000 100   0         0
10000      390697     1          967441
0          2572093    10000      830232
3953       144186     387        641860
10000      283720     0          0
```

FeelControlCtl.h

```
#if
!defined(AFX_FEELCONTROLCTL_H__78ACF773_5CC1_11D1_A868_00600
83A2742__INCLUDED_)
#define
AFX_FEELCONTROLCTL_H__78ACF773_5CC1_11D1_A868_0060083A2742__
INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

// FeelControlCtl.h : Declaration of the CFeelControlCtrl
ActiveX Control class.

/////////////////////////////////////////////////////////////
// CFeelControlCtrl : See FeelControlCtl.cpp for
implementation.

class CFeelControlCtrl : public COleControl
{
        DECLARE_DYNCREATE(CFeelControlCtrl)

// Constructor
public:
        CFeelControlCtrl();

// Overrides
        // ClassWizard generated virtual function
overrides
        //{{AFX_VIRTUAL(CFeelControlCtrl)
        public:
        virtual void OnDraw(CDC* pdc, const CRect&
rcBounds, const CRect& rcInvalid);
        virtual void DoPropExchange(CPropExchange* pPX);
        virtual void OnResetState();
        virtual DWORD GetControlFlags();
        //}}AFX_VIRTUAL

// Implementation
protected:
        ~CFeelControlCtrl();

        DECLARE_OLECREATE_EX(CFeelControlCtrl)     // Class
factory and guid
        DECLARE_OLETYPELIB(CFeelControlCtrl)       //
GetTypeInfo
```

```
        DECLARE_PROPPAGEIDS(CFeelControlCtrl)       //
Property page IDs
        DECLARE_OLECTLTYPE(CFeelControlCtrl)
        // Type name and misc status

// Message maps
        //{{AFX_MSG(CFeelControlCtrl)
                // NOTE - ClassWizard will add and
remove member functions here.
                //    DO NOT EDIT what you see in these
blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

// Dispatch maps
        //{{AFX_DISPATCH(CFeelControlCtrl)
        afx_msg BSTR GetEffect1();
        afx_msg void SetEffect1(LPCTSTR lpszNewValue);
        afx_msg BSTR GetEffect2();
        afx_msg void SetEffect2(LPCTSTR lpszNewValue);
        afx_msg BSTR GetEffect3();
        afx_msg void SetEffect3(LPCTSTR lpszNewValue);
        afx_msg BSTR GetEffect4();
        afx_msg void SetEffect4(LPCTSTR lpszNewValue);
        afx_msg BSTR GetEffect5();
        afx_msg void SetEffect5(LPCTSTR lpszNewValue);
        afx_msg BSTR GetEffect6();
        afx_msg void SetEffect6(LPCTSTR lpszNewValue);
        afx_msg long DoEffect(short effectNum);
        afx_msg long StopEffect(short effectNum);
        afx_msg void StopAll();
        afx_msg long SetEffect(short effectNum, LPCTSTR
effectParams);
        afx_msg long DoEnclosureEffect(short effectNum,
long left, long top, long right, long bottom);
        afx_msg long ApplyForce(long Xdir, long Ydir, long
Mag);
        afx_msg long StopForce();
        //}}AFX_DISPATCH
        DECLARE_DISPATCH_MAP()

// Event maps
        //{{AFX_EVENT(CFeelControlCtrl)
        //}}AFX_EVENT
        DECLARE_EVENT_MAP()

// Dispatch and event IDs
public:
        enum {
        //{{AFX_DISP_ID(CFeelControlCtrl)
        dispidEffect1 = 1L,
        dispidEffect2 = 2L,
        dispidEffect3 = 3L,
        dispidEffect4 = 4L,
        dispidEffect5 = 5L,
        dispidEffect6 = 6L,
        dispidDoEffect = 7L,
        dispidStopEffect = 8L,
        dispidStopAll = 9L,
        dispidSetEffect = 10L,
        dispidDoEnclosureEffect = 11L,
        dispidApplyForce = 12L,
        dispidStopForce = 13L,
        //}}AFX_DISP_ID
        };
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_FEELCONTROLCTL_H__78ACF773_5CC1_11D1_A868_00600
83A2742__INCLUDED)
```

FeelControlCtl.cpp

```
// FeelControlCtl.cpp : Implementation of the
CFeelControlCtrl ActiveX Control class.

#include "stdafx.h"
#include <objsafe.h>
#include <comcat.h>
#include "FeelControl.h"
#include "FeelControlCtl.h"
#include "FeelControlPpg.h"
#include "FeelForces.h"

HRESULT CreateComponentCategory( CATID catid, WCHAR*
catDescription );
HRESULT RegisterCLSIDInCategory( REFCLSID clsid, CATID catid
);
HRESULT UnregisterCLSIDInCategory( REFCLSID clsid, CATID
catid );

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CFeelControlCtrl, COleControl)

/////////////////////////////////////////////////////////////
// Message map

BEGIN_MESSAGE_MAP(CFeelControlCtrl, COleControl)
        //{{AFX_MSG_MAP(CFeelControlCtrl)
        // NOTE - ClassWizard will add and remove message
map entries
        //    DO NOT EDIT what you see in these blocks of
generated code !
        //}}AFX_MSG_MAP
        ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////
// Dispatch map
BEGIN_DISPATCH_MAP(CFeelControlCtrl, COleControl)
        //{{AFX_DISPATCH_MAP(CFeelControlCtrl)
        DISP_PROPERTY_EX(CFeelControlCtrl, "Effect1",
GetEffect1, SetEffect1, VT_BSTR)
        DISP_PROPERTY_EX(CFeelControlCtrl, "Effect2",
GetEffect2, SetEffect2, VT_BSTR)
        DISP_PROPERTY_EX(CFeelControlCtrl, "Effect3",
GetEffect3, SetEffect3, VT_BSTR)
        DISP_PROPERTY_EX(CFeelControlCtrl, "Effect4",
GetEffect4, SetEffect4, VT_BSTR)
        DISP_PROPERTY_EX(CFeelControlCtrl, "Effect5",
GetEffect5, SetEffect5, VT_BSTR)
        DISP_PROPERTY_EX(CFeelControlCtrl, "Effect6",
GetEffect6, SetEffect6, VT_BSTR)
        DISP_FUNCTION(CFeelControlCtrl, "DoEffect",
DoEffect, VT_I4, VTS_I2)
        DISP_FUNCTION(CFeelControlCtrl, "StopEffect",
StopEffect, VT_I4, VTS_I2)
        DISP_FUNCTION(CFeelControlCtrl, "StopAll",
StopAll, VT_EMPTY, VTS_NONE)
        DISP_FUNCTION(CFeelControlCtrl, "SetEffect",
SetEffect, VT_I4, VTS_I2 VTS_BSTR)
        DISP_FUNCTION(CFeelControlCtrl,
"DoEnclosureEffect", DoEnclosureEffect, VT_I4, VTS_I2 VTS_I4
VTS_I4 VTS_I4 VTS_I4)
        DISP_FUNCTION(CFeelControlCtrl, "ApplyForce",
ApplyForce, VT_I4, VTS_I4 VTS_I4 VTS_I4)
        DISP_FUNCTION(CFeelControlCtrl, "StopForce",
StopForce, VT_I4, VTS_NONE)
        //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()


/////////////////////////////////////////////////////////////
// Event map
BEGIN_EVENT_MAP(CFeelControlCtrl, COleControl)
        //{{AFX_EVENT_MAP(CFeelControlCtrl)
        // NOTE - ClassWizard will add and remove event
map entries
        //    DO NOT EDIT what you see in these blocks of
generated code !
        //}}AFX_EVENT_MAP
END_EVENT_MAP()


/////////////////////////////////////////////////////////////
// Property pages
// TODO: Add more property pages as needed.  Remember to
increase the count!
BEGIN_PROPPAGEIDS(CFeelControlCtrl, 1)
        PROPPAGEID(CFeelControlPropPage::guid)
END_PROPPAGEIDS(CFeelControlCtrl)


/////////////////////////////////////////////////////////////
// Initialize class factory and guid
IMPLEMENT_OLECREATE_EX(CFeelControlCtrl,
"FEELCONTROL.FeelControlCtrl.1",
        0x5dfdd466, 0x5b37, 0x11d1, 0xa8, 0x68, 0, 0x60,
0x8, 0x3a, 0x27, 0x42)


/////////////////////////////////////////////////////////////
// Type library ID and version
IMPLEMENT_OLETYPELIB(CFeelControlCtrl, _tlid, _wVerMajor,
_wVerMinor)


/////////////////////////////////////////////////////////////
// Interface IDs
const IID BASED_CODE IID_DFeelControl =
                { 0x78acf765, 0x5cc1, 0x11d1, { 0xa8,
0x68, 0, 0x60, 0x8, 0x3a, 0x27, 0x42 } };
const IID BASED_CODE IID_DFeelControlEvents =
                { 0x78acf766, 0x5cc1, 0x11d1, { 0xa8,
0x68, 0, 0x60, 0x8, 0x3a, 0x27, 0x42 } };


/////////////////////////////////////////////////////////////
// Control type information
static const DWORD BASED_CODE _dwFeelControlOleMisc =
        OLEMISC_INVISIBLEATRUNTIME |
        OLEMISC_ACTIVATEWHENVISIBLE |
          OLEMISC_IGNOREACTIVATEWHENVISIBLE |
          OLEMISC_SETCLIENTSITEFIRST |
          OLEMISC_INSIDEOUT |
          OLEMISC_CANTLINKINSIDE |
          OLEMISC_RECOMPOSEONRESIZE;

IMPLEMENT_OLECTLTYPE(CFeelControlCtrl, IDS_FEELCONTROL,
_dwFeelControlOleMisc)

/////////////////////////////////////////////////////////////
// CFeelControlCtrl::CFeelControlCtrlFactory::UpdateRegistry
-
// Adds or removes system registry entries for
CFeelControlCtrl

BOOL
CFeelControlCtrl::CFeelControlCtrlFactory::UpdateRegistry(BO
OL bRegister)
{
          // TODO: Verify that your control follows
apartment-model threading rules.
          // Refer to MFC TechNote 64 for more information.
          // If your control does not conform to the
apartment-model rules, then
          // you must modify the code below, changing the
6th parameter from
          // afxRegApartmentThreading to 0.
          if (bRegister)
          {
                    CreateComponentCategory( CATID_Control,
                                        L"Controls" );
                    RegisterCLSIDInCategory( m_clsid,
CATID_Control );

                    CreateComponentCategory(
CATID_SafeForInitializing,
L"Controls safely initializable from persistent data" );
                    RegisterCLSIDInCategory( m_clsid,
CATID_SafeForInitializing );

                    CreateComponentCategory(
CATID_SafeForScripting,
          L"Controls that are safely scriptable" );
                    RegisterCLSIDInCategory( m_clsid,
CATID_SafeForScripting );

                    CreateComponentCategory(
CATID_PersistsToPropertyBag,
          L"Support initialize via PersistPropertyBag" );
                    RegisterCLSIDInCategory( m_clsid,
CATID_PersistsToPropertyBag );

                    return AfxOleRegisterControlClass(
                              AfxGetInstanceHandle(),
                              m_clsid,
                              m_lpszProgID,
                              IDS_FEELCONTROL,
                              IDB_FEELCONTROL,
                              afxRegApartmentThreading,
                              _dwFeelControlOleMisc,
                              _tlid,
                              _wVerMajor,
                              _wVerMinor);
          }
          else
          {
                    UnregisterCLSIDInCategory( m_clsid,
CATID_Control );
                    UnregisterCLSIDInCategory( m_clsid,
CATID_PersistsToPropertyBag );
                    UnregisterCLSIDInCategory( m_clsid,
CATID_SafeForScripting );
                    UnregisterCLSIDInCategory( m_clsid,
CATID_SafeForInitializing );
                    return AfxOleUnregisterClass(m_clsid,
m_lpszProgID);
          }
}

/////////////////////////////////////////////////////////////
// CFeelControlCtrl::CFeelControlCtrl - Constructor

CFeelControlCtrl::CFeelControlCtrl()
{
          InitializeIIDs(&IID_DFeelControl,
&IID_DFeelControlEvents);

          // TODO: Initialize your control's instance data
here.
          FeelSetup( AfxGetInstanceHandle(),
AfxGetMainWnd()->m_hWnd );
}

/////////////////////////////////////////////////////////////
// CFeelControlCtrl::~CFeelControlCtrl - Destructor

CFeelControlCtrl::~CFeelControlCtrl()
{
          // TODO: Cleanup your control's instance data
here.
          FeelCleanup();
}
```

```
/////////////////////////////////////////////////////////////
// CFeelControlCtrl::OnDraw - Drawing function

void CFeelControlCtrl::OnDraw(
                              CDC* pdc, const CRect&
rcBounds, const CRect& rcInvalid)
{
          // TODO: Replace the following code with your own
drawing code.
          pdc->FillRect(rcBounds,
CBrush::FromHandle((HBRUSH)GetStockObject(WHITE_BRUSH)));
          pdc->Ellipse(rcBounds);
}

/////////////////////////////////////////////////////////////
// CFeelControlCtrl::DoPropExchange - Persistence support

void CFeelControlCtrl::DoPropExchange(CPropExchange* pPX)
{
          ExchangeVersion(pPX, MAKELONG(_wVerMinor,
_wVerMajor));
          COleControl::DoPropExchange(pPX);

          // TODO: Call PX_ functions for each persistent
custom property.
}

/////////////////////////////////////////////////////////////
// CFeelControlCtrl::GetControlFlags -
// Flags to customize MFC's implementation of ActiveX
controls.
// For information on using these flags, please see MFC
technical note
// #nnn, "Optimizing an ActiveX Control".
DWORD CFeelControlCtrl::GetControlFlags()
{
          DWORD dwFlags = COleControl::GetControlFlags();

          // The control can activate without creating a
window.
          // TODO: when writing the control's message
handlers, avoid using
          //                  the m_hWnd member variable
without first checking that its
          //                  value is non-NULL.
          dwFlags |= windowlessActivate;

          // The control can receive mouse notifications
when inactive.
          // TODO: if you write handlers for WM_SETCURSOR
and WM_MOUSEMOVE,
          //                  avoid using the m_hWnd member
variable without first
          //                  checking that its value is
non-NULL.
          dwFlags |= pointerInactive;
          return dwFlags;
}

/////////////////////////////////////////////////////////////
// CFeelControlCtrl::OnResetState - Reset control to default
state

void CFeelControlCtrl::OnResetState()
{
          COleControl::OnResetState();  // Resets defaults
found in DoPropExchange
          // TODO: Reset any other control state here.
}

/////////////////////////////////////////////////////////////
// CFeelControlCtrl message handlers

long CFeelControlCtrl::DoEffect(short effectNum)
{
          // TODO: Add your dispatch handler code here
          return FeelBeginEffect( effectNum );
}

long CFeelControlCtrl::StopEffect(short effectNum)
{
          // TODO: Add your dispatch handler code here
          return FeelEndEffect( effectNum );
}

void CFeelControlCtrl::StopAll()
{
          // TODO: Add your dispatch handler code here
          FeelEndAllEffects();
}

long CFeelControlCtrl::ApplyForce(long Xdir, long Ydir, long
Mag )
{
          return FeelBeginForce( Xdir, Ydir, Mag );
}

long CFeelControlCtrl::StopForce()
{
          return FeelEndForce();
}
```

```
long CFeelControlCtrl::DoEnclosureEffect(short effectNum,
long left, long top, long right, long bottom)
{
        // TODO: Add your dispatch handler code here
        return FeelEnclosureEffect( effectNum, left, top,
right, bottom );
}


long CFeelControlCtrl::SetEffect(short effectNum, LPCTSTR
effectParams)
{
        // TODO: Add your dispatch handler code here

        return 0;
}

BSTR CFeelControlCtrl::GetEffect1()
{
        CString strResult;
        // TODO: Add your property handler here

        return strResult.AllocSysString();
}

void CFeelControlCtrl::SetEffect1(LPCTSTR lpszNewValue)
{
        // TODO: Add your property handler here

        SetModifiedFlag();
}

BSTR CFeelControlCtrl::GetEffect2()
{
        CString strResult;
        // TODO: Add your property handler here

        return strResult.AllocSysString();
}

void CFeelControlCtrl::SetEffect2(LPCTSTR lpszNewValue)
{
        // TODO: Add your property handler here

        SetModifiedFlag();
}

BSTR CFeelControlCtrl::GetEffect3()
{
        CString strResult;
        // TODO: Add your property handler here

        return strResult.AllocSysString();
}

void CFeelControlCtrl::SetEffect3(LPCTSTR lpszNewValue)
{
        // TODO: Add your property handler here

        SetModifiedFlag();
}

BSTR CFeelControlCtrl::GetEffect4()
{
        CString strResult;
        // TODO: Add your property handler here

        return strResult.AllocSysString();
}

void CFeelControlCtrl::SetEffect4(LPCTSTR lpszNewValue)
{
        // TODO: Add your property handler here

        SetModifiedFlag();
}

BSTR CFeelControlCtrl::GetEffect5()
{
        CString strResult;
        // TODO: Add your property handler here

        return strResult.AllocSysString();
}

void CFeelControlCtrl::SetEffect5(LPCTSTR lpszNewValue)
{
        // TODO: Add your property handler here

        SetModifiedFlag();
}

BSTR CFeelControlCtrl::GetEffect6()
{
        CString strResult;
        // TODO: Add your property handler here

        return strResult.AllocSysString();
}

void CFeelControlCtrl::SetEffect6(LPCTSTR lpszNewValue)
{
        // TODO: Add your property handler here

        SetModifiedFlag();
}

HRESULT CreateComponentCategory( CATID catid, WCHAR*
catDescription )
{
        ICatRegister*       pcr = NULL;
        HRESULT                         hr  = S_OK;

        // Create an instance of the category manager
        hr = CoCreateInstance(
                CLSID_StdComponentCategoriesMgr,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_ICatRegister,
                (void**)&pcr
                );
        if (FAILED(hr))
                return hr;

        CATEGORYINFO catinfo;
        catinfo.catid = catid;
        catinfo.lcid = 0x0409; // English locale ID in hex

        int len = wcslen( catDescription );
        wcsncpy( catinfo.szDescription, catDescription,
len );
        catinfo.szDescription[len] = '\0';
        hr = pcr->RegisterCategories( 1, &catinfo );
        pcr->Release();
        return hr;
}

HRESULT RegisterCLSIDInCategory(REFCLSID clsid, CATID catid)
{
        ICatRegister* pcr = NULL;
        HRESULT    hr = S_OK;

        // Create an instance of the category manager
        hr = CoCreateInstance(
                CLSID_StdComponentCategoriesMgr,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_ICatRegister,
                (void**)&pcr
        );
        if (SUCCEEDED(hr))
        {
                CATID rgcatid[1];
                rgcatid[0] = catid;
                hr = pcr->RegisterClassImplCategories(
clsid, 1, rgcatid );
        }
        if ( pcr != NULL )
                pcr->Release();
        return hr;
}

HRESULT UnregisterCLSIDInCategory( REFCLSID clsid, CATID
catid )
{
        ICatRegister* pcr = NULL;
        HRESULT hr = S_OK;

        // Create an instance of the category manager
        hr = CoCreateInstance(
                CLSID_StdComponentCategoriesMgr,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_ICatRegister,
                (void**)&pcr
        );
        if (SUCCEEDED(hr))
        {
                CATID rgcatid[1];
                rgcatid[0] = catid;
                hr = pcr->UnRegisterClassImplCategories(
clsid, 1, rgcatid );
        }
        if ( pcr != NULL )
                pcr->Release();
        return hr;
}
```

FeelControlPpg.h

```
#if
!defined(AFX_FEELCONTROLPPG_H__78ACF775_5CC1_11D1_A868_00600
83A2742__INCLUDED_)
#define
AFX_FEELCONTROLPPG_H__78ACF775_5CC1_11D1_A868_0060083A2742__
INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
```

```
// FeelControlPpg.h : Declaration of the
CFeelControlPropPage property page class.

////////////////////////////////////////////////////////////////////////////
// CFeelControlPropPage : See FeelControlPpg.cpp.cpp for
implementation.

class CFeelControlPropPage : public COlePropertyPage
{
	DECLARE_DYNCREATE(CFeelControlPropPage)
	DECLARE_OLECREATE_EX(CFeelControlPropPage)

// Constructor
public:
	CFeelControlPropPage();

// Dialog Data
	//{{AFX_DATA(CFeelControlPropPage)
	enum { IDD = IDD_PROPPAGE_FEELCONTROL };
		// NOTE - ClassWizard will add data
members here.
		//    DO NOT EDIT what you see in these
blocks of generated code !
	//}}AFX_DATA

// Implementation
protected:
	virtual void DoDataExchange(CDataExchange* pDX);
// DDX/DDV support

// Message maps
protected:
	//{{AFX_MSG(CFeelControlPropPage)
		// NOTE - ClassWizard will add and
remove member functions here.
		//    DO NOT EDIT what you see in these
blocks of generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_FEELCONTROLPPG_H__78ACF775_5CC1_11D1_A868_00600
83A2742__INCLUDED)
```

FeelControlPpg.cpp

```
// FeelControlPpg.cpp : Implementation of the
CFeelControlPropPage property page class.

#include "stdafx.h"
#include "FeelControl.h"
#include "FeelControlPpg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CFeelControlPropPage, COlePropertyPage)

////////////////////////////////////////////////////////////////////////////
// Message map
BEGIN_MESSAGE_MAP(CFeelControlPropPage, COlePropertyPage)
	//{{AFX_MSG_MAP(CFeelControlPropPage)
	// NOTE - ClassWizard will add and remove message
map entries
	//    DO NOT EDIT what you see in these blocks of
generated code !
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// Initialize class factory and guid
IMPLEMENT_OLECREATE_EX(CFeelControlPropPage,
"FEELCONTROL.FeelControlPropPage.1",
	0x78acf767, 0x5cc1, 0x11d1, 0xa8, 0x68, 0, 0x60,
0x8, 0x3a, 0x27, 0x42)

////////////////////////////////////////////////////////////////////////////
//CFeelControlPropPage::CFeelControlPropPageFactory::UpdateR
egistry -
// Adds or removes system registry entries for
CFeelControlPropPage

BOOL
CFeelControlPropPage::CFeelControlPropPageFactory::UpdateReg
istry(BOOL bRegister)
{
	if (bRegister)
		return
AfxOleRegisterPropertyPageClass(AfxGetInstanceHandle(),
				m_clsid, IDS_FEELCONTROL_PPG);
	else
		return AfxOleUnregisterClass(m_clsid,
NULL);
}

////////////////////////////////////////////////////////////////////////////
// CFeelControlPropPage::CFeelControlPropPage - Constructor

CFeelControlPropPage::CFeelControlPropPage() :
	COlePropertyPage(IDD, IDS_FEELCONTROL_PPG_CAPTION)
{
	//{{AFX_DATA_INIT(CFeelControlPropPage)
	// NOTE: ClassWizard will add member
initialization here
	//    DO NOT EDIT what you see in these blocks of
generated code !
	//}}AFX_DATA_INIT
}

////////////////////////////////////////////////////////////////////////////
// CFeelControlPropPage::DoDataExchange - Moves data between
page and properties

void CFeelControlPropPage::DoDataExchange(CDataExchange*
pDX)
{
	//{{AFX_DATA_MAP(CFeelControlPropPage)
	// NOTE: ClassWizard will add DDP, DDX, and DDV
calls here
	//    DO NOT EDIT what you see in these blocks of
generated code !
	//}}AFX_DATA_MAP
	DDP_PostProcessing(pDX);
}
////////////////////////////////////////////////////////////////////////////
// CFeelControlPropPage message handlers
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by FeelControl.rc
#define IDS_FEELCONTROL              1
#define IDS_FEELCONTROL_PPG          2
#define IDS_FEELCONTROL_PPG_CAPTION  200
#define IDD_PROPPAGE_FEELCONTROL     200
#define IDB_FEELCONTROL              1
#define _APS_NEXT_RESOURCE_VALUE       201
#define _APS_NEXT_CONTROL_VALUE        201
#define _APS_NEXT_SYMED_VALUE          101
#define _APS_NEXT_COMMAND_VALUE        32768
```

StdAfx.h

```
#if
!defined(AFX_STDAFX_H__78ACF76A_5CC1_11D1_A868_0060083A2742_
_INCLUDED_)
#define
AFX_STDAFX_H__78ACF76A_5CC1_11D1_A868_0060083A2742__INCLUDED
_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// stdafx.h : include file for standard system include
files, or project specific include files that are used
frequently, but are changed infrequently

#define VC_EXTRALEAN          // Exclude rarely-used stuff
from Windows headers
#include <afxctl.h>      // MFC support for ActiveX Controls

// Delete the two includes below if you do not wish to use
the MFC database classes
#include <afxdb.h>            // MFC database classes
#include <afxdao.h>           // MFC DAO database classes

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.
#endif //
!defined(AFX_STDAFX_H__78ACF76A_5CC1_11D1_A868_0060083A2742_
_INCLUDED_)
```

StdAfx.cpp

```
// stdafx.cpp:source file that includes just the stnd
includes
// stdafx.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information
#include "stdafx.h"
```

APPENDIX C

Spring.htm ---Spring demo, FIG. 13a

```
<html><head>
<TITLE>Compress The Spring</TITLE>
<style>   .myStyle { font-family: verdana; color:white }
        </style>

<SCRIPT FOR=window EVENT="onload" LANGUAGE="JavaScript">
        document.onmousemove = compress;
</SCRIPT>

<SCRIPT language="JavaScript">
        var springForceFlag1 = false;
        var springForceFlag2 = false;
        var springForceFlag3 = false;

        var theSpringK1 = 10000;
        var theSpringK2 = 6000;
        var theSpringK3 = 2500;
        var previousY = 10000;

        function dospring(springDiv, springImg,
springForceFlag, theSpringK)
        {
                var xval = event.clientX;
                var yval = event.clientY;
                var minHeight;
                var minTop;

                // Check if we're touching spring
                if ( (xval >
(springDiv.offsetLeft+springImg.offsetLeft)) &&
                     (xval <
(springDiv.offsetLeft+springImg.offsetLeft+springImg.offsetW
idth)) )
                {
                        minHeight =
springDiv.offsetHeight/3;
                        minTop = springDiv.offsetTop +
springDiv.offsetHeight - minHeight;

                        if ( (yval >
springDiv.offsetTop) &&
                             (yval < minTop) )
                        { // in top 2/3 of spring
                                if (
!springForceFlag )
                                {
                                        if ( (yval
< springDiv.offsetTop+(springDiv.offsetHeight / 3)) ||

(previousY < springDiv.offsetTop+(springDiv.offsetHeight /
3)) )
                { // in start spring zone
        springForceFlag = true;
        DynamicObject.SetSpringK( theSpringK );
        DynamicObject.StartSpring( event.screenY-
(event.clientY-springDiv.style.pixelTop) );
                                        }
                                }
                                if (springForceFlag)
                                {
        springImg.style.top = (yval-springDiv.offsetTop) +
"px";
        springImg.style.height =
(springDiv.offsetTop+springDiv.offsetHeight-yval) + "px";
                                }
                        }
                        else
                        if ( yval >= minTop ) //
springDiv.offsetTop+springDiv.offsetHeight
                        { // below 1/3 of spring
                                if ( springForceFlag
)
                                {
        springImg.style.top = (minTop-springDiv.offsetTop)
+ "px"; // (springDiv.offsetHeight-1) + "px";
        springImg.style.height = minHeight + "px"; // 1 +
"px";
                                }
                        }
                        else
                        { // above spring
                                springImg.style.top
= 0 + "px";
        springImg.style.height = springDiv.offsetHeight +
"px";
                                if ( springForceFlag
)
                                {
        springForceFlag = false;
        DynamicObject.EndSpring();
                                }
                                }
                        }
                        else
                        { // left or right of spring
                                springImg.style.top = 0 +
"px";
                                springImg.style.height =
springDiv.offsetHeight + "px";
                                if ( springForceFlag )
                                {
                                        springForceFlag =
false;
        DynamicObject.EndSpring();
                                }
                        }
                        return springForceFlag;
        }

        function compress()
        {
                springForceFlag1 = dospring(springDiv1,
springImg1, springForceFlag1, theSpringK1);
                springImg1.style.width= "144px";
                springForceFlag2 = dospring(springDiv2,
springImg2, springForceFlag2, theSpringK2);
                springImg2.style.width= "96px";
                springForceFlag3 = dospring(springDiv3,
springImg3, springForceFlag3, theSpringK3);
                springImg3.style.width= "72px";
                previousY = event.clientY;
        }

</script>
</head>

<BODY TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#FF0000"
VLINK="#800080" ALINK="#0000FF"
BACKGROUND="images\background.jpg">
<CENTER><TABLE  >
<TR>
<TD></TD><TD>
<CENTER><IMG SRC="images\logo2_red.gif" HEIGHT=90
WIDTH=162></CENTER>
</TD><TD></TD>
<TD>
<CENTER><B><FONT SIZE=+2>Compress The
Springs</FONT></B></CENTER>
</TD><TD></TD>
<TD><IMG SRC="images\mouse.gif" HEIGHT=144 WIDTH=246></TD>
<TD></TD>
</TR>
</TABLE>
<CENTER>
<HR WIDTH="100%"><BR>
</CENTER>

<CENTER><DIV ID=DEBUGINFO> </DIV></CENTER>
<OBJECT ID="DynamicObject" WIDTH=0 HEIGHT=0
CLASSID="CLSID:EC296EE6-836C-11D1-A868-0060083A2742"
CODEBASE="DynamicControl.CAB#version=2,0,0,0">
</OBJECT>

<CENTER>
<div
        id=springDiv1
        style="position:absolute; left:120; top:210;
width:144; height:192; overflow:clip;"
>
        <img id=springImg1 style="position:absolute;
left:0; top:0;z-index:-1;" src="images\Spring1.gif">
</div>

<div
        id=springDiv2
        style="position:absolute; left:410; top:274;
width:96; height:128; overflow:clip;"
>
        <img id=springImg2 style="position:absolute;
left:0; top:0;z-index:-1;" src="images\Spring2.gif">
</div>

<div
        id=springDiv3
        style="position:absolute; left:645; top:306;
width:72; height:96; overflow:clip;"
>
        <img id=springImg3 style="position:absolute;
left:0; top:0;z-index:-1;" src="images\Spring3.gif">
</div>
</CENTER>

<div style="position:absolute; left:10; top:430;" >
<CENTER>
<BUTTON TYPE="BUTTON" TITLE="Back"
        LANGUAGE="JavaScript"
onmouseup="window.navigate('wa2.htm')"
>
Back
</BUTTON>
<BUTTON TYPE="BUTTON" TITLE="Next"
        LANGUAGE="JavaScript"
onmouseup="window.navigate('pop.htm')"
>
```

```
Next
</BUTTON>
</CENTER>

<CENTER><P>
<HR WIDTH="100%"><BR><I><FONT SIZE=-1>feelit@immerse.com<BR>
Copyright (c) 1996-1998, Immersion
Corporation</FONT></I></P></CENTER>
</div>

</body>
</html>
```

---

pop.htm --- Ball popping demo, FIG. 13b

```
<html><head>
<TITLE>Pop The Ball</TITLE>
<style>   .myStyle { font-family: verdana; color:white }
          </style>

<SCRIPT language=VBScript>
//function window_onload()
//        initialize()
//end function
</SCRIPT>

<SCRIPT FOR=window EVENT="onload" LANGUAGE="JavaScript">
          document.onmousemove = compress;
</SCRIPT>

<SCRIPT language="JavaScript">
          var nerfForceFlag = false;
          var nerfPoppedFlag = false;
          var poppingFactor = 0.60;
          var theBallK = 10000;
          var previousY = 10000;

          function compress()
          {
                    var xval = event.clientX;
                    var yval = event.clientY;

                    //
                    // Nerf
                    //
                    if ( ! nerfPoppedFlag )
                    {
                       // Check if we're touching the nerf
                              if ( (xval >
(nerfDiv.offsetLeft+nerfImg.offsetLeft)) &&
                                   (xval <
(nerfDiv.offsetLeft+nerfImg.offsetLeft+nerfImg.offsetWidth))
)
                              {
                                        if ( (yval >
nerfDiv.offsetTop) &&
                                             (yval <
(nerfDiv.offsetTop+nerfDiv.offsetHeight)) )
                                        {
                                                  if ( (yval
< nerfDiv.offsetTop+(nerfDiv.offsetHeight / 3)) ||
 (previousY < nerfDiv.offsetTop+(nerfDiv.offsetHeight / 3))
)
                                                  {  // in
start spring zone
          if ( ! nerfForceFlag )
          {
          nerfForceFlag = true;
          DynamicObject.SetSpringK( theBallK );
          DynamicObject.StartSpring( event.screenY-
(event.clientY-nerfDiv.style.pixelTop) );
          }
                                                  }
                                                  if (
nerfForceFlag ) {
          if ( yval >
(nerfDiv.offsetTop+(nerfDiv.offsetHeight*poppingFactor)) )
          {
          PopSound.Run();
          DynamicObject.EndSpring();
          //DynamicObject.Pop();
          nerfImg.style.height = (51) + "px";      // 51 =
nerfpop.gif height
          nerfImg.style.top = (nerfDiv.offsetHeight-(51)) +
"px";
          // Change Image
          nerfImg.src = "images\\nerfpop.gif";
          nerfPoppedFlag = true;
          nerfForceFlag = false;
          }
          else
          {
          nerfImg.style.top = (yval-nerfDiv.offsetTop) +
"px";
          nerfImg.style.height =
(nerfDiv.offsetTop+nerfDiv.offsetHeight-yval) + "px";
          }
                                                  }
                                        }
                                        else
                                        if ( yval >=
(nerfDiv.offsetTop+nerfDiv.offsetHeight) )
                                        {
                                                  if (
nerfForceFlag )
                                                  {
          nerfImg.style.top = (nerfDiv.offsetHeight-1) +
"px";
          nerfImg.style.height = 1 + "px";
                                                  }
                                        }
                                        else
                                        {
          nerfImg.style.top = 0 + "px";
          nerfImg.style.height = nerfDiv.offsetHeight +
"px";
          nerfForceFlag = false;
          DynamicObject.EndSpring();
                                        }
                              }
                              else
                              {
                                        nerfImg.style.top =
0 + "px";
                                        nerfImg.style.height
= nerfDiv.offsetHeight + "px";
                                        if ( nerfForceFlag )
                                        {
          nerfForceFlag = false;
          DynamicObject.EndSpring();
                                        }
                              }
                              nerfImg.style.width= "174px";
                    }

                    previousY = yval;
          }

          function restoreBall()
          {
                    BoingSound.Run();
                    // Change Image
                    nerfImg.src = "images\\nerf.gif";
                    // Change Location and Size
                    nerfImg.style.top = 0 + "px";
                    nerfImg.style.height =
nerfDiv.offsetHeight + "px";
                    nerfImg.style.width= "174px";
                    // Change Pop Flag
                    nerfPoppedFlag = false;
          }
</script>
</head>

<body
          bgcolor=ffffff
>
<BODY TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#FF0000"
VLINK="#800080" ALINK="#0000FF"
BACKGROUND="images\background.jpg">
<CENTER><TABLE  >
<TR>
<TD></TD><TD>
<CENTER><IMG SRC="images\logo2_red.gif" HEIGHT=90
WIDTH=162></CENTER>
</TD><TD></TD>
<TD>
<CENTER><B><FONT SIZE=+2>Pop The Ball</FONT></B></CENTER>
</TD><TD></TD>
<TD><IMG SRC="images\mouse.gif" HEIGHT=144 WIDTH=246></TD>
<TD></TD>
</TR>
</TABLE>
<CENTER>
<HR WIDTH="100%"><BR>
</CENTER>

<CENTER><DIV ID=DEBUGINFO> </DIV></CENTER>
<OBJECT ID="DynamicObject" WIDTH=0 HEIGHT=0
CLASSID="CLSID:EC296EE6-836C-11D1-A868-0060083A2742"
CODEBASE="DynamicControl.CAB#version=2,0,0,0">
</OBJECT>

<OBJECT ID="PopSound" WIDTH=0 HEIGHT=0
CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A">
          <PARAM NAME="ShowControls" VALUE="0">
          <PARAM NAME="ShowDisplay"  VALUE="0">
          <PARAM NAME="AutoStart"    VALUE="0">
          <PARAM NAME="AutoRewind"   VALUE="1">
          <PARAM NAME="FileName"
VALUE="sounds\pop3.wav">
</OBJECT>

<OBJECT ID="BoingSound" WIDTH=0 HEIGHT=0
CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A">
          <PARAM NAME="ShowControls" VALUE="0">
          <PARAM NAME="ShowDisplay"  VALUE="0">
          <PARAM NAME="AutoStart"    VALUE="0">
          <PARAM NAME="AutoRewind"   VALUE="1">
          <PARAM NAME="FileName"
VALUE="sounds\boing.wav">
</OBJECT>
```

```
<div
          id=nerfDiv
          style="position:absolute; left:360; top:210;
width:174; height:192; overflow:clip;"
>
          <img id=nerfImg style="position:absolute; left:0;
top:0;z-index:-1;" src="images\nerf.gif">
</div>

<BUTTON TYPE="BUTTON" TITLE="Inflate Ball"
                    STYLE="position:absolute; left:240;
top:290;"
                    LANGUAGE="JavaScript"
onmouseup="restoreBall()"
>
Inflate Ball
</BUTTON>

<div style="position:absolute; left:10; top:440;" >
<CENTER>
<BUTTON TYPE="BUTTON" TITLE="Back"
          LANGUAGE="JavaScript"
onmouseup="window.navigate('spring.htm')"
>
Back
</BUTTON>
<BUTTON TYPE="BUTTON" TITLE="Next"
          LANGUAGE="JavaScript"
onmouseup="window.navigate('ball.htm')"
>
Next
</BUTTON>
</CENTER>
<CENTER><P>
<HR WIDTH="100%"><BR><I><FONT SIZE=-1>feelit@immerse.com<BR>
Copyright (c) 1996-1998, Immersion
Corporation</FONT></I></P></CENTER>
</div>

</body>
</html>
```

---

ball.htm ---Dynamic ball demo, FIG. 14

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Developer Studio">
<META HTTP-EQUIV="Content-Type" content="text/html;
charset=iso-8859-1">
<TITLE>Bounce The Ball</TITLE>
</HEAD>

<BODY TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#FF0000"
VLINK="#800080" ALINK="#0000FF"
BACKGROUND="images\background.jpg">
<CENTER><TABLE  >
<TR>
<TD></TD>
<TD><CENTER><IMG SRC="images\logo2_red.gif" HEIGHT=90
WIDTH=162></CENTER>
</TD><TD></TD><TD>
<CENTER><B><FONT SIZE=+2>Bounce The Ball</FONT></B></CENTER>
</TD><TD></TD>
<TD><IMG SRC="images\mouse.gif" HEIGHT=144 WIDTH=246></TD>
<TD></TD>
</TR>
</TABLE><CENTER>
<HR WIDTH="100%"><BR>
</CENTER>

<!--<BODY style="background-image:
url(images\BallBackground.jpg); background-repeat: no-
repeat;">

<!--  Here are the images -->
<CENTER><DIV id=DEBUGINFO> </DIV></CENTER>

<OBJECT ID="DynamicObject" WIDTH=0 HEIGHT=0
CLASSID="CLSID:EC296EE6-836C-11D1-A868-0060083A2742"
CODEBASE="DynamicControl.CAB#version=2,0,0,0">
</OBJECT>

<OBJECT ID="BonkSound" WIDTH=0 HEIGHT=0
CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A">
          <PARAM NAME="ShowControls" VALUE="0">
          <PARAM NAME="ShowDisplay"  VALUE="0">
          <PARAM NAME="AutoStart"    VALUE="0">
          <PARAM NAME="AutoRewind"   VALUE="1">
          <PARAM NAME="FileName"
VALUE="sounds\bonk.wav">
</OBJECT>

<IMG id=CourtImg src="images\court.jpg" style="position:
absolute; top: 195; left:225; z-index: -1">
<IMG id=Ball1Img src="images\ball.gif"  style="position:
absolute; top: 200; left:230; z-index: 1">
<!-- BallImage Size is 100x100 -->

<div style="position:absolute; left:10; top:535;" >
<CENTER>
<BUTTON TYPE="BUTTON" TITLE="Back"
          LANGUAGE="JavaScript"
onmouseup="window.navigate('pop.htm')"
>
Back
</BUTTON>
<BUTTON TYPE="BUTTON" TITLE="Next"
          LANGUAGE="JavaScript"
onmouseup="window.navigate('demo.html')"
>
Next
</BUTTON>
</CENTER>
<P>
<CENTER><HR WIDTH="100%"><BR><I><FONT SIZE=-
1>feelit@immerse.com<BR>
Copyright (c) 1996-1998, Immersion
Corporation</FONT></I></CENTER>
</div>

<SCRIPT FOR=window EVENT="onload" LANGUAGE="JavaScript">
// Initialize -- start the ticker!
          document.onmousemove = doMouseMove;
          Ball1Radius = 0.5 * Ball1Img.offsetWidth;
          Ball1Img.style.height = Ball1Img.style.width;
          DynamicObject.StartBall();
          tick();
</SCRIPT>

<SCRIPT language=JavaScript>
var tickTimeout;
tickTimeout = 1;
var oldTime = new Date();
var n = 0;

var PlaygroundLeft, PlaygroundTop;
var PlaygroundWidth, PlaygroundHeight;
PlaygroundLeft = 230;
PlaygroundTop = 200;
PlaygroundWidth  = 362;//390;
PlaygroundHeight = 208;//290;

var Ball1Mass, Ball1K;
var Ball1Xp, Ball1Yp, Ball1Xpp, Ball1Ypp;
Ball1Mass = 10;
Ball1K = 0.5;
Ball1Xp = 0;
Ball1Yp = 0;
Ball1Xpp = 0;
Ball1Ypp = 0;

var ForceFlag;
ForceFlag = false;

// Periodically calls itself
function tick()
{
          moveBalls();
          window.setTimeout("tick();", tickTimeout,
"JavaScript" );
}

// Perform a Ball movement simulation
function moveBalls()
{
          // Calc dynamics for Ball1 during this time step
          Ball1Xp += Ball1Xpp;
          Ball1Yp += Ball1Ypp;
          Ball1Img.style.pixelLeft += Ball1Xp;
          Ball1Img.style.pixelTop  += Ball1Yp;

          // WALL COLLISION DETECTION
          if ( Ball1Img.style.pixelLeft < PlaygroundLeft )
          {
                    BonkSound.Run();
                    Ball1Img.style.pixelLeft =
PlaygroundLeft;
                    Ball1Xp = -Ball1Xp*0.75;
          }
          else
          if ( Ball1Img.style.pixelLeft >
(PlaygroundLeft+PlaygroundWidth) )
          {
                    BonkSound.Run();
                    Ball1Img.style.pixelLeft =
(PlaygroundLeft+PlaygroundWidth);
                    Ball1Xp = -Ball1Xp*0.75;
          }
          if ( Ball1Img.style.pixelTop < PlaygroundTop )
          {
                    BonkSound.Run();
                    Ball1Img.style.pixelTop = PlaygroundTop;
                    Ball1Yp = -Ball1Yp*0.75;
          }
          else
          if ( Ball1Img.style.pixelTop >
(PlaygroundTop+PlaygroundHeight) )
          {
                    BonkSound.Run();
                    Ball1Img.style.pixelTop =
(PlaygroundTop+PlaygroundHeight);
                    Ball1Yp = -Ball1Yp*0.75;
```

```
        }
//      CalcLoopRate();
}

function CalcLoopRate()
{
        var rate;
        n = n+1;
        if ( n == 100 )
        {
                newTime = new Date();
                rate = newTime.getTime() -
oldTime.getTime();
                oldTime = newTime;
//              DEBUGINFO.innerHTML = (1000/(rate/n));
                n=0;
        }
}

// When the mouse moves, do bounds checking and possibly
alter the Ball's X/Ypp function doMouseMove()
{
        var xc, yc, w, mag;

        // Ball 1
        xc = event.clientX -
(Ball1Img.offsetLeft+Ball1Radius);
        yc = event.clientY - (Ball1Img.offsetTop
+Ball1Radius);
        w = xc*xc + yc*yc;

        // Are we inside the Ball?
        if ( (w < (Ball1Radius*Ball1Radius)) )
        {
                ForceFlag = true;
                mag = -((Ball1Radius/Math.sqrt(w)-1) *
Ball1K);
                DynamicObject.ApplyForce( xc, yc,
mag*50000 );
                Ball1Xpp = mag * xc / Ball1Mass;
                Ball1Ypp = mag * yc / Ball1Mass;
        }
        else
        {
                // use the flag to prevent this from
being called lots of times!
                if ( ForceFlag == true )
                {
                        // No, so there will be no
applied force
                        Ball1Xpp = 0;
                        Ball1Ypp = 0;
                        DynamicObject.EndForce();
                        ForceFlag = false;
                }
        }
        var xabs, yabs;
        xabs = event.screenX - (event.offsetX -
Ball1Img.style.pixelLeft );
        yabs = event.screenY - (event.offsetY -
Ball1Img.style.pixelTop );
//      DynamicObject.ChangeBallPos( xabs, yabs );
}

</SCRIPT>

</BODY>
</HTML>
```

pendulum.htm -- pendulum demo, FIG. 15

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Cart-Pendulum FEELit Mouse Demonstration</TITLE>
</HEAD>

<BODY BGCOLOR=BLACK TEXT=RED>
<DIV STYLE="font-size: 12 pt; font-family: Verdana, Arial,
Helvetica">
<H3 ID=myHead>Cart-Pendulum Game</H3>
</DIV>

<CENTER><DIV id=DEBUGINFO> </DIV></CENTER>
<HR>
<OBJECT ID=Serra WIDTH=0 HEIGHT=0
CLASSID="CLSID:EC296EE6-836C-11D1-A868-0060083A2742"
CODEBASE="DynamicControl.CAB#version=1,0,0,1">
</OBJECT>

<OBJECT ID=Cart
STYLE="Position:absolute; WIDTH:625; HEIGHT:425; top:10;
left:70; Z-INDEX: 1"
CLASSID="CLSID:369303C2-D7AC-11d0-89D5-00A0C90833E6"
>
        <PARAM NAME="CoordinateSystem"          VALUE="1">
        <PARAM NAME="MouseEventsEnabled"        VALUE="1">
</OBJECT>

<OBJECT ID=Pendulum
STYLE="Position:absolute; WIDTH:625; HEIGHT:425; top:10;
left:70; Z-INDEX: 2"
CLASSID="CLSID:369303C2-D7AC-11d0-89D5-00A0C90833E6">
        <PARAM NAME="CoordinateSystem"          VALUE="1">
        <PARAM NAME="MouseEventsEnabled"        VALUE="1">
</OBJECT>

<SCRIPT language=VBScript>
function window_onload()
        initialize()
end function
</SCRIPT>

<SCRIPT LANGUAGE="Javascript">

var cartX, cartXp, cartXpp;
var cartY, cartWidth, cartHeight;
var cartMass;
cartXpp = 0;
cartXp  = 0;
cartX = 300;
cartY = 200;
cartWidth  = 50;
cartHeight = 30;
cartMass = 1;

var cartK = 0.4;
var forceFactor = 4000;

var trackX, trackY, trackWidth, trackHeight;
trackX = 0;
trackY = cartHeight/2;
trackWidth = Cart.style.pixelWidth;
trackHeight = 10;

var linkWidth, linkHeight, plumbDiameter;
linkWidth = 10;
linkLength = 100;
plumbDiameter = 30;

var pendT, pendTp, pendTpp;
var pendMass, pendInertia;
pendT = 170;
pendTp =  0;
pendTpp = 0;
pendMass = 1;
pendInertia = 1;

var myOffsetX, myOffsetY;
myOffsetX = 415;
myOffsetY = 250;

var g = 9.81;
var friction = 0.5;
var forceX = 0;
var forcePosMax = 50;

var oldTime = new Date();
var period;

var lastMouseX = 0;
var lastMouseY = 0;
var buttonFlag = false;
var forceFlag  = false;

var lib = Cart.Library;         // This sets up the
DirectAnimation Library for
                                //    DrawingSurface
operations.

//
// Initialize our scripts
function initialize()
{
        CreateScene();
        TransformScene();
        tick();
}

// Tick() is performed every tick...
function tick()
{
        CalcLoopRate();
        Simulate();
        TransformScene();
        window.setTimeout("tick();", 1, "JavaScript" );
}

// CalcLoopRate
//  Figure # of seconds since last call to this function.
//      Stores value in global period.
function CalcLoopRate()
{
        newTime = new Date();
        period = ( newTime.getTime() - oldTime.getTime() )
/ 1000;
        oldTime = newTime;
        period *= 2;            // Scale to integrate faster
(make time fly!)
}
```

```
//  Creates the scene
function CreateScene()
{
        var ds;
        // Draw the cart and Track
        Cart.SetIdentity();
        ds = Cart.DrawingSurface;
                // The Cart
                ds.FillColor( lib.blue );
                ds.Rect( -(cartWidth/2), -
(cartHeight/2), cartWidth, cartHeight );
                // The Track
                ds.FillColor( lib.green );
                ds.Rect( (trackX-300-(trackWidth/2)),
trackY, trackWidth*3, trackHeight );
        Cart.DrawingSurface = ds;
        // Draw the linkage and plumb-bob
        ds = Pendulum.DrawingSurface;
                // The Linkage
                ds.FillColor(lib.ColorRgb255(255,0,0));
                ds.Rect( -(linkWidth/2), 0, linkWidth,
linkLength );
                // The plumb-bob
        ds.FillColor(lib.ColorRgb255(200,200,255));
                ds.Oval( -(plumbDiameter/2),
(linkLength-(plumbDiameter/2)), plumbDiameter, plumbDiameter
);
        Pendulum.DrawingSurface = ds;
}

// Transform scene
function TransformScene()
{
        Cart.SetIdentity();
        Cart.Translate( cartX-myOffsetX, cartY-myOffsetY,
0 );
        Pendulum.SetIdentity();
        Pendulum.Rotate( 0, 0, -pendT );
        Pendulum.Translate( cartX-myOffsetX, cartY-
myOffsetY, 0 );
}

// Performs dynamic simulation
function Simulate()
{
        var oldTRad             = pendT*(Math.PI/180);
        var cosTRad = Math.cos(oldTRad);
        var sinTRad = Math.sin(oldTRad);
        var oldTpp              = pendTpp;
        var oldXpp              = cartXpp;

        // Check interaction force
        if ( forceFlag )
        {
                forceX = (lastMouseX - cartX) * cartK;
                DEBUGINFO.innerHTML=forceX;
                if ( forceX > forcePosMax )
                        forceX = forcePosMax;
                else
                if ( forceX < -forcePosMax)
                        forceX = -forcePosMax;
        }
        else
                forceX = 0;

        // Move the cart
        cartXpp = ( forceX -
(pendMass*linkLength*oldTpp*cosTRad) +
(pendMass*linkLength*pendTp*pendTp*sinTRad) -
(friction*cartXp) ) / (pendMass+cartMass);
        cartXp += cartXpp*period;
        cartX  += cartXp*period;
        if ( (cartX-(cartWidth/2)-103) < trackX )
        {
                cartX   = trackX+(cartWidth/2)+103;
                cartXp = 0;//-cartXp;
                cartXpp = 0;
        }
        else
        if ( (cartX+(cartWidth/2)-103) >
(trackX+trackWidth) )
        {
                cartX   = (trackX+trackWidth-
(cartWidth/2))+103;
                cartXp = 0;//-cartXp;
                cartXpp = 0;
        }

        // Move the pendulum
        pendTpp = - ( (g*sinTRad) + (oldXpp*cosTRad) ) / (
(pendInertia/(pendMass*linkLength))+linkLength);
        pendTp += pendTpp*period;
        pendT  = (oldTRad + pendTp*period) *
(180/Math.PI);
//      DEBUGINFO.innerHTML= pendTpp + " *** " + pendTp +
" *** " + oldTRad + " *** " + pendT + " *** Per:" + period;

        // Apply Force
        if ( forceFlag == true )
        {
                Serra.ApplyForce( 1, 0,
forceX*forceFactor );
        }
        else
                Serra.ApplyForce( 1, 0, 0 );
}

// DoMouseMove
function doMouseMove(button,clientX,clientY)
{
        if ( buttonFlag )
        {
//              if ( (clientY>(cartY-(cartHeight/2)+10))
&& (clientY<(cartY+(cartHeight/2)+10)) )
//              {
                        forceFlag = true;
                        lastMouseX = clientX;
//              }
//              else
//                      forceFlag = false;
        }
        else
                forceFlag = false;
}

// doMouseDown
function doMouseDown(button, clientX, clientY)
{
        // Check if it's the left mouse button
        if ( button == 1 )
        {
                // Check if we're inside the box
                if ( (clientX>(cartX-(cartWidth/2))) &&
(clientX<(cartX+(cartWidth/2))) && (clientY>(cartY-
(cartHeight/2))) && (clientY<(cartY+(cartHeight/2))) )
                {
                        buttonFlag = true;
                        forceFlag  = true;
                        lastMouseX = clientX;
                }
        }
}

// doMouseUp
function doMouseUp(button, clientX, clientY )
{
        // Check if it's the left mouse button
        if ( button == 1 )
        {
                buttonFlag = false;
                forceFlag = false;
        }
}

</SCRIPT>
<SCRIPT FOR=Cart EVENT=onmousedown(button,shift,x,y)
LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="car";
        doMouseDown(button,x+70-40,y+10-110+25);
</SCRIPT>
<SCRIPT FOR=Cart EVENT=onmouseup(button,shift,x,y)
LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="car";
        doMouseUp(button,x+70-40,y+10-110+25);
</SCRIPT>
<SCRIPT FOR=Cart EVENT=onmousemove(button,shift,x,y)
LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="car";
        doMouseMove(button,x+70-40,y+10-110+25);
</SCRIPT>
<SCRIPT FOR=Pendulum EVENT=onmousedown(button,shift,x,y)
LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="pen";
        doMouseDown(button,x+70-40,y+10-110+25);
</SCRIPT>
<SCRIPT FOR=Pendulum EVENT=onmouseup(button, shift,x,y)
LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="pen";
        doMouseUp(button,x+70-40,y+10-110+25);
</SCRIPT>
<SCRIPT FOR=Pendulum EVENT=onmousemove(button,shift,x,y)
LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="pen";
        doMouseMove(button,x+70-40,y+10-110+25);
</SCRIPT>
<SCRIPT FOR=document EVENT=onmousedown LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="doc";
        doMouseDown( event.button, event.clientX+40,
event.clientY-70 );
</SCRIPT>
<SCRIPT FOR=document EVENT=onmouseup LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="doc";
        doMouseUp( event.button, event.clientX+40,
event.clientY-70);
</SCRIPT>
<SCRIPT FOR=document EVENT=onmousemove LANGUAGE="JScript">
//      DEBUGINFO.innerHTML="doc";
        doMouseMove( event.clientY, event.clientX+40,
event.clientY-70 );
</SCRIPT>

<SCRIPT FOR=myHead EVENT=onmousedown LANGUAGE="JScript">
        tick();
</SCRIPT>
```

```
</BODY>
</HTML>
```

Remaining listings in Appendix C are used for all the demos in Appendix C

DynamicControl.odl

```
// DynamicControl.odl : type library source for ActiveX
Control project.
// This file will be processed by the Make Type Library
(mktyplib) tool to
// produce the type library (DynamicControl.tlb) that will
become a resource in DynamicControl.ocx.

#include <olectl.h>
#include <idispids.h>

[ uuid(EC296EE3-836C-11D1-A868-0060083A2742), version(1.0),
  helpfile("DynamicControl.hlp"),
  helpstring("DynamicControl ActiveX Control module"),
  control ]
library DYNAMICCONTROLLib
{
        importlib(STDOLE_TLB);
        importlib(STDTYPE_TLB);

        //  Primary dispatch interface for
CDynamicControlCtrl

        [ uuid(EC296EE4-836C-11D1-A868-0060083A2742),
          helpstring("Dispatch interface for
DynamicControl Control"), hidden ]
        dispinterface _DDynamicControl
        {
                properties:
                        // NOTE - ClassWizard will maintain
property information here.
                        //    Use extreme caution when editing
this section.
                        //{{AFX_ODL_PROP(CDynamicControlCtrl)
                        //}}AFX_ODL_PROP

                methods:
                        // NOTE - ClassWizard will maintain
method information here.
                        //    Use extreme caution when editing
this section.
                        //{{AFX_ODL_METHOD(CDynamicControlCtrl)
                        [id(1)] long ApplyForce(long Xdir, long
Ydir, long Mag);
                        [id(2)] long EndForce();
                        [id(3)] long StartBall();
                        [id(4)] long EndBall();
                        [id(5)] long ChangeBallPos(long leftVal,
long topVal);
                        [id(6)] long StartSpring(long topVal);
                        [id(7)] long EndSpring();
                        [id(8)] long StartNerf();
                        [id(9)] long EndNerf();
                        [id(10)] long ChangeNerfRect(long left,
long top, long width, long height);
                        [id(11)] long SetSpringK(long theK);
                        [id(12)] long Pop();
                        //}}AFX_ODL_METHOD

                        [id(DISPID_ABOUTBOX)] void AboutBox();
        };

        //  Event dispatch interface for
CDynamicControlCtrl

        [ uuid(EC296EE5-836C-11D1-A868-0060083A2742),
          helpstring("Event interface for DynamicControl
Control") ]
        dispinterface _DDynamicControlEvents
        {
                properties:
                        //  Event interface has no properties

                methods:
                        // NOTE - ClassWizard will maintain
event information here.
                        //    Use extreme caution when editing
this section.
                        //{{AFX_ODL_EVENT(CDynamicControlCtrl)
                        //}}AFX_ODL_EVENT
        };

        //  Class infoımation for CDynamicContıolCtrl

        [ uuid(EC296EE6-836C-11D1-A868-0060083A2742),
          helpstring("DynamicControl Control"), control ]
        coclass DynamicControl
        {
                        [default] dispinterface
_DDynamicControl;
                        [default, source] dispinterface
_DDynamicControlEvents;
        };

        //{{AFX_APPEND_ODL}}
        //}}AFX_APPEND_ODL}}
};
```

DynamicControl.inf

```
[version]
    signature="$CHICAGO$"
    AdvancedINF=2.0
 [Add.Code]
    DynamicControl.ocx=DynamicControl.ocx
    msvcrt.dll=msvcrt.dll
    mfc42.dll=mfc42.dll
    olepro32.dll=olepro32.dll
 [DynamicControl.ocx]
    file-win32-x86=thiscab
    clsid={EC296EE6-836C-11D1-A868-0060083A2742}
    FileVersion=1,0,0,1
    RegisterServer=yes
 [msvcrt.dll]
    FileVersion=4,20,0,6164
    hook=mfc42installer
 [mfc42.dll]
    FileVersion=4,2,0,6256
    hook=mfc42installer
 [olepro32.dll]
    FileVersion=4,2,0,6068
    hook=mfc42installer
 [mfc42installer]
    file-win32-
x86=http://activex.microsoft.com/controls/vc/mfc42.cab
    run=%EXTRACT_DIR%\mfc42.exe
```

DynamicControl.def

```
; DynamicControl.def : Declares the module parameters.

LIBRARY      "DYNAMICCONTROL.OCX"

EXPORTS
        DllCanUnloadNow      @1 PRIVATE
        DllGetClassObject    @2 PRIVATE
        DllRegisterServer    @3 PRIVATE
        DllUnregisterServer @4 PRIVATE
```

DynamicControl.rc

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "1 TYPELIB ""DynamicControl.tlb""\r\n"
    "\0"
END
```

```
#endif    // APSTUDIO_INVOKED

#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
VS_VERSION_INFO VERSIONINFO
 FILEVERSION 2,0,0,0
 PRODUCTVERSION 2,0,0,0
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x2L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "CompanyName", "Immersion Corporation\0"
            VALUE "FileDescription", "DynamicControl ActiveX
Control Module\0"
            VALUE "FileVersion", "2, 0, 0, 0\0"
            VALUE "InternalName", "DYNAMICCONTROL\0"
            VALUE "LegalCopyright", "Copyright (C) 1998\0"
            VALUE "OriginalFilename", "DYNAMICCONTROL.OCX\0"
            VALUE "ProductName", "DynamicControl ActiveX
Control Module\0"
            VALUE "ProductVersion", "2, 0, 0, 0\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC

/////////////////////////////////////////////////////////////////////////////
// Icon
// Icon with lowest ID value placed first to ensure
application icon
// remains consistent on all systems.
IDI_ABOUTDLL            ICON    DISCARDABLE
"DynamicControl.ico"

/////////////////////////////////////////////////////////////////////////////
// Bitmap
IDB_DYNAMICCONTROL      BITMAP  DISCARDABLE
"DynamicControlCtl.bmp"

/////////////////////////////////////////////////////////////////////////////
// Dialog
IDD_ABOUTBOX_DYNAMICCONTROL DIALOG DISCARDABLE  34, 22, 260,
55
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About DynamicControl Control"
FONT 8, "MS Sans Serif"
BEGIN
    ICON            IDI_ABOUTDLL,IDC_STATIC,10,10,20,20
    LTEXT           "DynamicControl Control, Version
1.0",IDC_STATIC,40,10,
                    170,8
    LTEXT           "Copyright (C) 1998, Immersion
Corporation",IDC_STATIC,
                    40,25,170,8
    DEFPUSHBUTTON   "OK",IDOK,221,7,32,14,WS_GROUP
END

IDD_PROPPAGE_DYNAMICCONTROL DIALOG DISCARDABLE  0, 0, 250,
62
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "TODO: Place controls to manipulate
properties of DynamicControl Control on this dialog.",
                    IDC_STATIC,7,25,229,16
END

/////////////////////////////////////////////////////////////////////////////
// DESIGNINFO
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX_DYNAMICCONTROL, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 253
        TOPMARGIN, 7
        BOTTOMMARGIN, 48
    END

    IDD_PROPPAGE_DYNAMICCONTROL, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 243
        TOPMARGIN, 7
        BOTTOMMARGIN, 55
    END
END
#endif    // APSTUDIO_INVOKED

/////////////////////////////////////////////////////////////////////////////
// String Table
STRINGTABLE DISCARDABLE
BEGIN
    IDS_DYNAMICCONTROL      "DynamicControl Control"
    IDS_DYNAMICCONTROL_PPG  "DynamicControl Property Page"
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_DYNAMICCONTROL_PPG_CAPTION "General"
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////

#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
// Generated from the TEXTINCLUDE 3 resource.
//
1 TYPELIB "DynamicControl.tlb"

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

DynamicControl.h

```
#if
!defined(AFX_DYNAMICCONTROL_H__EC296EEC_836C_11D1_A868_00600
83A2742__INCLUDED_)
#define
AFX_DYNAMICCONTROL_H__EC296EEC_836C_11D1_A868_0060083A2742__
INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

// DynamicControl.h : main header file for
DYNAMICCONTROL.DLL

#if !defined( __AFXCTL_H__ )
        #error include 'afxctl.h' before including this
file
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CDynamicControlApp : See DynamicControl.cpp for
implementation.

class CDynamicControlApp : public COleControlModule
{
public:
        BOOL InitInstance();
        int ExitInstance();
};

extern const GUID CDECL _tlid;
extern const WORD _wVerMajor;
extern const WORD _wVerMinor;

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_DYNAMICCONTROL_H__EC296EEC_836C_11D1_A868_00600
83A2742__INCLUDED)
```

DynamicControl.cpp

```
// DynamicControl.cpp : Implementation of CDynamicControlApp
and DLL registration.

#include "stdafx.h"
#include "DynamicControl.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CDynamicControlApp NEAR theApp;

const GUID CDECL BASED_CODE _tlid =
                { 0xec296ee3, 0x836c, 0x11d1, { 0xa8,
0x68, 0, 0x60, 0x8, 0x3a, 0x27, 0x42 } };
const WORD _wVerMajor = 1;
const WORD _wVerMinor = 0;
```

```
/////////////////////////////////////////////////////////////
// CDynamicControlApp::InitInstance - DLL initialization

BOOL CDynamicControlApp::InitInstance()
{
		BOOL bInit = COleControlModule::InitInstance();

		if (bInit)
		{
				// TODO: Add your own module
initialization code here.
		}
		return bInit;
}

/////////////////////////////////////////////////////////////
// CDynamicControlApp::ExitInstance - DLL termination

int CDynamicControlApp::ExitInstance()
{
		// TODO: Add your own module termination code
here.
		return COleControlModule::ExitInstance();
}

/////////////////////////////////////////////////////////////
// DllRegisterServer - Adds entries to the system registry

STDAPI DllRegisterServer(void)
{
		AFX_MANAGE_STATE(_afxModuleAddrThis);

		if (!AfxOleRegisterTypeLib(AfxGetInstanceHandle(),
_tlid))
				return
ResultFromScode(SELFREG_E_TYPELIB);

		if (!COleObjectFactoryEx::UpdateRegistryAll(TRUE))
				return ResultFromScode(SELFREG_E_CLASS);
		return NOERROR;
}

/////////////////////////////////////////////////////////////
// DllUnregisterServer - Removes entries from the system
registry

STDAPI DllUnregisterServer(void)
{
		AFX_MANAGE_STATE(_afxModuleAddrThis);

		if (!AfxOleUnregisterTypeLib(_tlid, _wVerMajor,
_wVerMinor))
				return
ResultFromScode(SELFREG_E_TYPELIB);

		if
(!COleObjectFactoryEx::UpdateRegistryAll(FALSE))
				return ResultFromScode(SELFREG_E_CLASS);

		return NOERROR;
}
```

DynamicControlCtl.h

```
#if
!defined(AFX_DYNAMICCONTROLCTL_H__EC296EF4_836C_11D1_A868_00
60083A2742__INCLUDED_)
#define
AFX_DYNAMICCONTROLCTL_H__EC296EF4_836C_11D1_A868_0060083A274
2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

// DynamicControlCtl.h : Declaration of the
CDynamicControlCtrl ActiveX Control class.

/////////////////////////////////////////////////////////////
// CDynamicControlCtrl : See DynamicControlCtl.cpp for
implementation.

class CDynamicControlCtrl : public COleControl
{
		DECLARE_DYNCREATE(CDynamicControlCtrl)

// Constructor
public:
		CDynamicControlCtrl();

// Overrides
		// ClassWizard generated virtual function
overrides
		//{{AFX_VIRTUAL(CDynamicControlCtrl)
		public:
		virtual void OnDraw(CDC* pdc, const CRect&
rcBounds, const CRect& rcInvalid);
		virtual void DoPropExchange(CPropExchange* pPX);
		virtual void OnResetState();
		virtual DWORD GetControlFlags();
		//}}AFX_VIRTUAL

// Implementation
protected:
		~CDynamicControlCtrl();

		DECLARE_OLECREATE_EX(CDynamicControlCtrl)	//
Class factory and guid
		DECLARE_OLETYPELIB(CDynamicControlCtrl)		//
GetTypeInfo
		DECLARE_PROPPAGEIDS(CDynamicControlCtrl)	//
Property page IDs
		DECLARE_OLECTLTYPE(CDynamicControlCtrl)
		// Type name and misc status

// Message maps
		//{{AFX_MSG(CDynamicControlCtrl)
				// NOTE - ClassWizard will add and
remove member functions here.
				//    DO NOT EDIT what you see in these
blocks of generated code !
		//}}AFX_MSG
		DECLARE_MESSAGE_MAP()

// Dispatch maps
		//{{AFX_DISPATCH(CDynamicControlCtrl)
		afx_msg long ApplyForce(long Xdir, long Ydir, long
Mag);
		afx_msg long EndForce();
		afx_msg long StartBall();
		afx_msg long EndBall();
		afx_msg long ChangeBallPos(long leftVal, long
topVal);
		afx_msg long StartSpring(long topVal);
		afx_msg long EndSpring();
		afx_msg long StartNerf();
		afx_msg long EndNerf();
		afx_msg long ChangeNerfRect(long left, long top,
long width, long height);
		afx_msg long SetSpringK(long theK);
		afx_msg long Pop();
		//}}AFX_DISPATCH
		DECLARE_DISPATCH_MAP()

		afx_msg void AboutBox();

// Event maps
		//{{AFX_EVENT(CDynamicControlCtrl)
		//}}AFX_EVENT
		DECLARE_EVENT_MAP()

// Dispatch and event IDs
public:
		enum {
		//{{AFX_DISP_ID(CDynamicControlCtrl)
		dispidApplyForce = 1L,
		dispidEndForce = 2L,
		dispidStartBall = 3L,
		dispidEndBall = 4L,
		dispidChangeBallPos = 5L,
		dispidStartSpring = 6L,
		dispidEndSpring = 7L,
		dispidStartNerf = 8L,
		dispidEndNerf = 9L,
		dispidChangeNerfRect = 10L,
		dispidSetSpringK = 11L,
		dispidPop = 12L,
		//}}AFX_DISP_ID
		};
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_DYNAMICCONTROLCTL_H__EC296EF4_836C_11D1_A868_00
60083A2742__INCLUDED)
```

DynamicControlCtl.cpp

```
// DynamicControlCtl.cpp : Implementation of the
CDynamicControlCtrl ActiveX Control class.

#include "stdafx.h"
#include <objsafe.h>
#include <comcat.h>
#include "DynamicControl.h"
#include "DynamicControlCtl.h"
#include "DynamicControlPpg.h"

#include "DynamicForces.h"

HRESULT CreateComponentCategory( CATID catid, WCHAR*
catDescription );
HRESULT RegisterCLSIDInCategory( REFCLSID clsid, CATID catid
);
HRESULT UnregisterCLSIDInCategory( REFCLSID clsid, CATID
catid );

#ifdef _DEBUG
#define new DEBUG_NEW
```

```
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CDynamicControlCtrl, COleControl)

////////////////////////////////////////////////////////////
// Message map
BEGIN_MESSAGE_MAP(CDynamicControlCtrl, COleControl)
        //{{AFX_MSG_MAP(CDynamicControlCtrl)
        // NOTE - ClassWizard will add and remove message
map entries
        //    DO NOT EDIT what you see in these blocks of
generated code !
        //}}AFX_MSG_MAP
        ON_OLEVERB(AFX_IDS_VERB_EDIT, OnEdit)
        ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////
// Dispatch map
BEGIN_DISPATCH_MAP(CDynamicControlCtrl, COleControl)
        //{{AFX_DISPATCH_MAP(CDynamicControlCtrl)
        DISP_FUNCTION(CDynamicControlCtrl, "ApplyForce",
ApplyForce, VT_I4, VTS_I4 VTS_I4 VTS_I4)
        DISP_FUNCTION(CDynamicControlCtrl, "EndForce",
EndForce, VT_I4, VTS_NONE)
        DISP_FUNCTION(CDynamicControlCtrl, "StartBall",
StartBall, VT_I4, VTS_NONE)
        DISP_FUNCTION(CDynamicControlCtrl, "EndBall",
EndBall, VT_I4, VTS_NONE)
        DISP_FUNCTION(CDynamicControlCtrl,
"ChangeBallPos", ChangeBallPos, VT_I4, VTS_I4 VTS_I4)
        DISP_FUNCTION(CDynamicControlCtrl, "StartSpring",
StartSpring, VT_I4, VTS_I4)
        DISP_FUNCTION(CDynamicControlCtrl, "EndSpring",
EndSpring, VT_I4, VTS_NONE)
        DISP_FUNCTION(CDynamicControlCtrl, "StartNerf",
StartNerf, VT_I4, VTS_NONE)
        DISP_FUNCTION(CDynamicControlCtrl, "EndNerf",
EndNerf, VT_I4, VTS_NONE)
        DISP_FUNCTION(CDynamicControlCtrl,
"ChangeNerfRect", ChangeNerfRect, VT_I4, VTS_I4 VTS_I4
VTS_I4 VTS_I4)
        DISP_FUNCTION(CDynamicControlCtrl, "SetSpringK",
SetSpringK, VT_I4, VTS_I4)
        DISP_FUNCTION(CDynamicControlCtrl, "Pop", Pop,
VT_I4, VTS_NONE)
        //}}AFX_DISPATCH_MAP
        DISP_FUNCTION_ID(CDynamicControlCtrl, "AboutBox",
DISPID_ABOUTBOX, AboutBox, VT_EMPTY, VTS_NONE)
END_DISPATCH_MAP()

////////////////////////////////////////////////////////////
// Event map
BEGIN_EVENT_MAP(CDynamicControlCtrl, COleControl)
        //{{AFX_EVENT_MAP(CDynamicControlCtrl)
        // NOTE - ClassWizard will add and remove event
map entries
        //    DO NOT EDIT what you see in these blocks of
generated code !
        //}}AFX_EVENT_MAP
END_EVENT_MAP()

////////////////////////////////////////////////////////////
// Property pages
// TODO: Add more property pages as needed.  Remember to
increase the count!
BEGIN_PROPPAGEIDS(CDynamicControlCtrl, 1)
        PROPPAGEID(CDynamicControlPropPage::guid)
END_PROPPAGEIDS(CDynamicControlCtrl)

////////////////////////////////////////////////////////////
// Initialize class factory and guid
IMPLEMENT_OLECREATE_EX(CDynamicControlCtrl,
"DYNAMICCONTROL.DynamicControlCtrl.1",
        0xec296ee6, 0x836c, 0x11d1, 0xa8, 0x68, 0, 0x60,
0x8, 0x3a, 0x27, 0x42)

////////////////////////////////////////////////////////////
// Type library ID and version
IMPLEMENT_OLETYPELIB(CDynamicControlCtrl, _tlid, _wVerMajor,
_wVerMinor)

////////////////////////////////////////////////////////////
// Interface IDs
const IID BASED_CODE IID_DDynamicControl =
        { 0xec296ee4, 0x836c, 0x11d1, { 0xa8, 0x68, 0,
0x60, 0x8, 0x3a, 0x27, 0x42 } };
const IID BASED_CODE IID_DDynamicControlEvents =
        { 0xec296ee5, 0x836c, 0x11d1, { 0xa8, 0x68, 0,
0x60, 0x8, 0x3a, 0x27, 0x42 } };

////////////////////////////////////////////////////////////
// Control type information
static const DWORD BASED_CODE _dwDynamicControlOleMisc =
        OLEMISC_INVISIBLEATRUNTIME |
        OLEMISC_SETCLIENTSITEFIRST |
        OLEMISC_INSIDEOUT |
        OLEMISC_CANTLINKINSIDE |
        OLEMISC_RECOMPOSEONRESIZE;
```

```
IMPLEMENT_OLECTLTYPE(CDynamicControlCtrl,
IDS_DYNAMICCONTROL, _dwDynamicControlOleMisc)

////////////////////////////////////////////////////////////
CDynamicControlCtrl::CDynamicControlCtrlFactory::UpdateRegis
try -
// Adds or removes system registry entries for
CDynamicControlCtrl

BOOL
CDynamicControlCtrl::CDynamicControlCtrlFactory::UpdateRegis
try(BOOL bRegister)
{
        // TODO: Verify that your control follows
apartment-model threading rules.
        // Refer to MFC TechNote 64 for more information.
        // If your control does not conform to the
apartment-model rules, then
        // you must modify the code below, changing the
6th parameter from
        // afxRegInsertable | afxRegApartmentThreading to
afxRegInsertable.

        if (bRegister)
        {
                CreateComponentCategory( CATID_Control,
                                L"Controls" );
                RegisterCLSIDInCategory( m_clsid,
CATID_Control );

                CreateComponentCategory(
CATID_SafeForInitializing,
        L"Controls safely initializable from persistent
data" );
                RegisterCLSIDInCategory( m_clsid,
CATID_SafeForInitializing );

                CreateComponentCategory(
CATID_SafeForScripting,
        L"Controls that are safely scriptable" );
                RegisterCLSIDInCategory( m_clsid,
CATID_SafeForScripting );

                CreateComponentCategory(
CATID_PersistsToPropertyBag,
        L"Support initialize via PersistPropertyBag" );
                RegisterCLSIDInCategory( m_clsid,
CATID_PersistsToPropertyBag );
                return AfxOleRegisterControlClass(
                        AfxGetInstanceHandle(),
                        m_clsid,
                        m_lpszProgID,
                        IDS_DYNAMICCONTROL,
                        IDB_DYNAMICCONTROL,
                        afxRegInsertable |
afxRegApartmentThreading,
                        _dwDynamicControlOleMisc,
                        _tlid,
                        _wVerMajor,
                        _wVerMinor);
        }
        else
        {
                UnregisterCLSIDInCategory( m_clsid,
CATID_Control );
                UnregisterCLSIDInCategory( m_clsid,
CATID_PersistsToPropertyBag );
                UnregisterCLSIDInCategory( m_clsid,
CATID_SafeForScripting );
                UnregisterCLSIDInCategory( m_clsid,
CATID_SafeForInitializing );
                return AfxOleUnregisterClass(m_clsid,
m_lpszProgID);
        }
}

////////////////////////////////////////////////////////////
// CDynamicControlCtrl::CDynamicControlCtrl - Constructor

CDynamicControlCtrl::CDynamicControlCtrl()
{
        InitializeIIDs(&IID_DDynamicControl,
&IID_DDynamicControlEvents);

        // TODO: Initialize your control's instance data
here.
        FeelSetup( AfxGetInstanceHandle(),
AfxGetMainWnd()->m_hWnd );
}

////////////////////////////////////////////////////////////
// CDynamicControlCtrl::~CDynamicControlCtrl - Destructor

CDynamicControlCtrl::~CDynamicControlCtrl()
{
        // TODO: Cleanup your control's instance data
here.
        FeelCleanup();
}

////////////////////////////////////////////////////////////
// CDynamicControlCtrl::OnDraw - Drawing function
```

```
void CDynamicControlCtrl::OnDraw(
                        CDC* pdc, const CRect& rcBounds, const
CRect& rcInvalid)
{
        // TODO: Replace the following code with your own
drawing code.
        pdc->FillRect(rcBounds,
CBrush::FromHandle((HBRUSH)GetStockObject(WHITE_BRUSH)));
        pdc->Ellipse(rcBounds);
}

/////////////////////////////////////////////////////////////
// CDynamicControlCtrl::DoPropExchange - Persistence support

void CDynamicControlCtrl::DoPropExchange(CPropExchange* pPX)
{
        ExchangeVersion(pPX, MAKELONG(_wVerMinor,
_wVerMajor));
        COleControl::DoPropExchange(pPX);
        // TODO: Call PX_ functions for each persistent
custom property.

}

/////////////////////////////////////////////////////////////
// CDynamicControlCtrl::GetControlFlags -
// Flags to customize MFC's implementation of ActiveX
controls.
//
// For information on using these flags, please see MFC
technical note
// #nnn, "Optimizing an ActiveX Control".
DWORD CDynamicControlCtrl::GetControlFlags()
{
        DWORD dwFlags = COleControl::GetControlFlags();


        // The control can activate without creating a
window.
        // TODO: when writing the control's message
handlers, avoid using
        //        the m_hWnd member variable without first
checking that its value is non-NULL.
        dwFlags |= windowlessActivate;
        return dwFlags;
}

/////////////////////////////////////////////////////////////
// CDynamicControlCtrl::OnResetState - Reset control to
default state

void CDynamicControlCtrl::OnResetState()
{
        COleControl::OnResetState();  // Resets defaults
found in DoPropExchange
        // TODO: Reset any other control state here.
}

/////////////////////////////////////////////////////////////
// CDynamicControlCtrl::AboutBox - Display an "About" box to
the user

void CDynamicControlCtrl::AboutBox()
{
        CDialog dlgAbout(IDD_ABOUTBOX_DYNAMICCONTROL);
        dlgAbout.DoModal();
}

/////////////////////////////////////////////////////////////
// CDynamicControlCtrl message handlers

long CDynamicControlCtrl::ApplyForce(long Xdir, long Ydir,
long Mag)
{
        return FeelBeginForce( Xdir, Ydir, Mag );
}

long CDynamicControlCtrl::EndForce()
{
        return FeelEndForce();
}


HRESULT CreateComponentCategory( CATID catid, WCHAR*
catDescription )
{
        ICatRegister*       pcr = NULL;
        HRESULT                 hr  = S_OK;
        // Create an instance of the category manager
        hr = CoCreateInstance(
                CLSID_StdComponentCategoriesMgr,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_ICatRegister,
                (void**)&pcr
                 );
        if (FAILED(hr))
                return hr;

        CATEGORYINFO catinfo;
        catinfo.catid = catid;
        catinfo.lcid = 0x0409; // English locale ID in hex
```

```
        int len = wcslen( catDescription );
        wcsncpy( catinfo.szDescription, catDescription,
len );
        catinfo.szDescription[len] = '\0';
        hr = pcr->RegisterCategories( 1, &catinfo );
        pcr->Release();

        return hr;
}

HRESULT RegisterCLSIDInCategory(REFCLSID clsid, CATID catid)
{
        ICatRegister* pcr = NULL;
        HRESULT   hr = S_OK;

        // Create an instance of the category manager
        hr = CoCreateInstance(
                CLSID_StdComponentCategoriesMgr,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_ICatRegister,
                (void**)&pcr
        );
        if (SUCCEEDED(hr))
        {
                CATID rgcatid[1];
                rgcatid[0] = catid;
                hr = pcr->RegisterClassImplCategories(
clsid, 1, rgcatid );
        }
        if ( pcr != NULL )
                pcr->Release();
        return hr;
}

HRESULT UnregisterCLSIDInCategory( REFCLSID clsid, CATID
catid )
{
        ICatRegister* pcr = NULL;
        HRESULT hr = S_OK;
        // Create an instance of the category manager
        hr = CoCreateInstance(
                CLSID_StdComponentCategoriesMgr,
                NULL,
                CLSCTX_INPROC_SERVER,
                IID_ICatRegister,
                (void**)&pcr
        );
        if (SUCCEEDED(hr))
        {
                CATID rgcatid[1];
                rgcatid[0] = catid;
                hr = pcr->UnRegisterClassImplCategories(
clsid, 1, rgcatid );
        }
        if ( pcr != NULL )
                pcr->Release();
        return hr;
}

long CDynamicControlCtrl::StartBall()
{
        return FeelBeginBall();
}
long CDynamicControlCtrl::EndBall()
{
        return FeelEndBall();
}
long CDynamicControlCtrl::ChangeBallPos(long leftVal, long
topVal)
{
        return FeelChangeBallLocation( leftVal, topVal );
}
long CDynamicControlCtrl::StartSpring( long topVal)
{
        return FeelBeginSpring( topVal );
}
long CDynamicControlCtrl::EndSpring()
{
        return FeelEndSpring();
}
long CDynamicControlCtrl::StartNerf()
{
        return FeelBeginNerf();
}
long CDynamicControlCtrl::EndNerf()
{
        return FeelEndNerf();
}
long CDynamicControlCtrl::ChangeNerfRect(long left, long
top, long width, long height)
{
        return FeelChangeNerfRect( left, top, width,
height );
}
long CDynamicControlCtrl::SetSpringK(long theK)
{
        return FeelSetSpring( theK );
}
long CDynamicControlCtrl::Pop()
{
        return FeelPop();
}
```

DynamicControlPpg.h

```
#if
!defined(AFX_DYNAMICCONTROLPPG_H__EC296EF6_836C_11D1_A868_00
60083A2742__INCLUDED_)
#define
AFX_DYNAMICCONTROLPPG_H__EC296EF6_836C_11D1_A868_0060083A274
2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

// DynamicControlPpg.h : Declaration of the
CDynamicControlPropPage property page class.

////////////////////////////////////////////////////////////
// CDynamicControlPropPage : See DynamicControlPpg.cpp.cpp
for implementation.

class CDynamicControlPropPage : public COlePropertyPage
{
          DECLARE_DYNCREATE(CDynamicControlPropPage)
          DECLARE_OLECREATE_EX(CDynamicControlPropPage)

// Constructor
public:
          CDynamicControlPropPage();

// Dialog Data
          //{{AFX_DATA(CDynamicControlPropPage)
          enum { IDD = IDD_PROPPAGE_DYNAMICCONTROL };
                    // NOTE - ClassWizard will add data
members here.
          //}}AFX_DATA

// Implementation
protected:
          virtual void DoDataExchange(CDataExchange* pDX);
// DDX/DDV support

// Message maps
protected:
          //{{AFX_MSG(CDynamicControlPropPage)
                    // NOTE - ClassWizard will add and
remove member functions here.
                    //    DO NOT EDIT what you see in these
blocks of generated code !
          //}}AFX_MSG
          DECLARE_MESSAGE_MAP()

};
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.
#endif //
!defined(AFX_DYNAMICCONTROLPPG_H__EC296EF6_836C_11D1_A868_00
60083A2742__INCLUDED)
```

DynamicControlPpg.cpp

```
// DynamicControlPpg.cpp : Implementation of the
CDynamicControlPropPage property page class.

#include "stdafx.h"
#include "DynamicControl.h"
#include "DynamicControlPpg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CDynamicControlPropPage,
COlePropertyPage)

////////////////////////////////////////////////////////////
// Message map

BEGIN_MESSAGE_MAP(CDynamicControlPropPage, COlePropertyPage)
          //{{AFX_MSG_MAP(CDynamicControlPropPage)
          // NOTE - ClassWizard will add and remove message
map entries
          //    DO NOT EDIT what you see in these blocks of
generated code !
          //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////
// Initialize class factory and guid

IMPLEMENT_OLECREATE_EX(CDynamicControlPropPage,
"DYNAMICCONTROL.DynamicControlPropPage.1",
          0xec296ee7, 0x836c, 0x11d1, 0xa8, 0x68, 0, 0x60,
0x8, 0x3a, 0x27, 0x42)

////////////////////////////////////////////////////////////
//
CDynamicControlPropPage::CDynamicControlPropPageFactory::Upd
ateRegistry -
// Adds or removes system registry entries for
CDynamicControlPropPage

BOOL
CDynamicControlPropPage::CDynamicControlPropPageFactory::Upd
ateRegistry(BOOL bRegister)
{
          if (bRegister)
                    return
AfxOleRegisterPropertyPageClass(AfxGetInstanceHandle(),
                              m_clsid,
IDS_DYNAMICCONTROL_PPG);
          else
                    return AfxOleUnregisterClass(m_clsid,
NULL);
}

////////////////////////////////////////////////////////////
// CDynamicControlPropPage::CDynamicControlPropPage -
Constructor

CDynamicControlPropPage::CDynamicControlPropPage() :
          COlePropertyPage(IDD,
IDS_DYNAMICCONTROL_PPG_CAPTION)
{
          //{{AFX_DATA_INIT(CDynamicControlPropPage)
          // NOTE: ClassWizard will add member
initialization here
          //    DO NOT EDIT what you see in these blocks of
generated code !
          //}}AFX_DATA_INIT
}

////////////////////////////////////////////////////////////
// CDynamicControlPropPage::DoDataExchange - Moves data
between page and properties

void CDynamicControlPropPage::DoDataExchange(CDataExchange*
pDX)
{
          //{{AFX_DATA_MAP(CDynamicControlPropPage)
          // NOTE: ClassWizard will add DDP, DDX, and DDV
calls here
          //    DO NOT EDIT what you see in these blocks of
generated code !
          //}}AFX_DATA_MAP
          DDP_PostProcessing(pDX);
}

////////////////////////////////////////////////
// CDynamicControlPropPage message handlers
```

DynamicForces.h

```
/**********************************************************
 * FeelControl
 * (c) 1997 Immersion Corporation
 * FILE
 *        FeelForces.h
 * DESCRIPTION
 *        Provide methods for doing force-feedback with the
ForceClasses, giving the FeelControl some guts...
 */

#ifndef    __FEELFORCES_H
#define __FEELFORCES_H

BOOL FeelSetup( HINSTANCE hInst, HWND hWnd );
BOOL FeelCleanup( void );

long FeelBeginForce( long Xdir, long Ydir, long Mag );
long FeelEndForce( void );

long FeelBeginBall( void );
long FeelEndBall( void );
long FeelChangeBallLocation( long left, long top );

long FeelBeginSpring( long top );
long FeelEndSpring( void );
long FeelSetSpring( long springK );

long FeelBeginNerf( void );
long FeelEndNerf( void );
long FeelChangeNerfRect( long left, long top, long width,
long height );

long FeelPop( void );

#endif    __FEELFORCES_H
```

DynamicForces.cpp

```
/***********************************************************
 * FeelControl
 * (c) 1997-1998 Immersion Corporation
* FILE
 *                    FeelForces.cpp
* DESCRIPTION
 *                    Provide methods for doing force-feedback
with the ForceClasses, giving the DynamicControl some
guts...
*/

#include "stdafx.h"
#include "DynamicForces.h"
#include "ForceFeelitMouse.h"
#include "ForceEffect.h"
#include "ForcePeriodic.h"
#include "ForceDamper.h"
#include "ForceEllipse.h"
#include "ForceCondition.h"
#include "ForceConstant.h"
#include "ForceEnclosure.h"
#include "ForceSpring.h"
#include <stdio.h>

// GLOBAL VARIABLES
CForceFeelitMouse*  gMouse      = NULL;
CForceConstant*               gForce      = NULL;
CForceEllipse*                gBall       = NULL;
CForceSpring*                 gSpring = NULL;
CForcePeriodic*               gPop1       = NULL;
CForcePeriodic*               gPop2       = NULL;
CForceEllipse*                gNerf       = NULL;

/*
 *  Globals for our params
 */

// Ball1.gif is 100x100
#define BALL_IMAGE_HEIGHT      100
#define BALL_IMAGE_WIDTH       100
#define BALL_WALL_WIDTH                       (BALL_IMAGE_WIDTH/4)
#define   BALL_STIFFNESS                      (8000)

#define NERF_IMAGE_HEIGHT      100
#define NERF_IMAGE_WIDTH       100
#define NERF_WALL_WIDTH                       (NERF_IMAGE_WIDTH/4)
#define   NERF_STIFFNESS                      (8000)

// Updown pop
#define POP1_DURATION                         300
#define POP1_PERIOD                           468
#define POP1_MAGNITUDE                        10000
const 'POINT POP1_DIRECTION = { 0, 1 };

// Leftright pop
#define POP2_DURATION                         300
#define POP2_PERIOD                           242
#define POP2_MAGNITUDE                        4000
const POINT POP2_DIRECTION = { 1, 1 };

BOOL FeelSetup( HINSTANCE hInst, HWND hWnd )
{
          BOOL      success;
          RECT      ballRect = { 0, 0, BALL_IMAGE_HEIGHT,
BALL_IMAGE_WIDTH };
          RECT      nerfRect = { 0, 0, NERF_IMAGE_HEIGHT,
NERF_IMAGE_WIDTH };

          // Set up the Mouse
          gMouse = new CForceFeelitMouse();
          if ( ! gMouse ) goto FS_Err;
          success = gMouse->Initialize( hInst, hWnd );
          if ( ! success ) goto FS_Err;

          // Set up the Force
          gForce = new CForceConstant();
          if ( ! gForce ) goto FS_Err;
          success = gForce->Initialize(gMouse);
          if ( ! success ) goto FS_Err;

          // Set up the Ball
          gBall = new CForceEllipse();
          if ( ! gBall ) goto FS_Err;
          success = gBall->Initialize(
              gMouse,
              &ballRect,
              BALL_STIFFNESS,
          //FORCE_ELLIPSE_DEFAULT_STIFFNESS,
              BALL_WALL_WIDTH,
              FORCE_ELLIPSE_DEFAULT_SATURATION,
              FEELIT_FSTIFF_OUTBOUNDANYWALL,
              FORCE_ELLIPSE_DEFAULT_CLIPPING_MASK,
              NULL
          );
          if ( ! success ) goto FS_Err;

          // Set up the Spring
          gSpring = new CForceSpring();
          if ( ! gSpring ) goto FS_Err;
          success = gSpring->Initialize(
                    gMouse,
                    10000,
          //FORCE_SPRING_DEFAULT_STIFFNESS,
                    FORCE_SPRING_DEFAULT_SATURATION,
                    0,
          //FORCE_SPRING_DEFAULT_DEADBAND,
                    FORCE_EFFECT_AXIS_Y,
                    FORCE_SPRING_DEFAULT_CENTER_POINT
          );
          if ( ! success ) goto FS_Err;

          // Set up the Nerf
          gNerf = new CForceEllipse();
          if ( ! gNerf ) goto FS_Err;
          success = gNerf->Initialize(
              gMouse,
              &nerfRect,
              -NERF_STIFFNESS,
          //FORCE_ELLIPSE_DEFAULT_STIFFNESS,
              NERF_WALL_WIDTH,
              FORCE_ELLIPSE_DEFAULT_SATURATION,
              FEELIT_FSTIFF_OUTBOUNDANYWALL,
              FORCE_ELLIPSE_DEFAULT_CLIPPING_MASK,
              NULL
          );
          if ( ! success ) goto FS_Err;

          // Set up the Pop#1
          gPop1 = new CForcePeriodic(GUID_Feelit_Square);
          if ( ! gPop1 ) goto FS_Err;
          success = gPop1->Initialize(
        gMouse,
        POP2_MAGNITUDE,
        POP2_PERIOD,
        POP2_DURATION,
                    POP2_DIRECTION.x,
                    POP2_DIRECTION.y,
        FORCE_PERIODIC_DEFAULT_OFFSET,
        FORCE_PERIODIC_DEFAULT_PHASE
          );
          if ( ! success ) goto FS_Err;

          // Set up the Pop#2
          gPop2 = new CForcePeriodic(GUID_Feelit_Square);
          if ( ! gPop2 ) goto FS_Err;
          success = gPop2->Initialize(
        gMouse,
        POP2_MAGNITUDE,
        POP2_PERIOD,
        POP2_DURATION,
                    POP2_DIRECTION.x,
                    POP2_DIRECTION.y,
        FORCE_PERIODIC_DEFAULT_OFFSET,
        FORCE_PERIODIC_DEFAULT_PHASE
          );
          if ( ! success ) goto FS_Err;

          // We're okay!
          return TRUE;

FS_Err:
          // There were some problems... let's cleanup and
declare ourselves dead!
          FeelCleanup();
          return FALSE;
}

BOOL FeelCleanup( void )
{
          if ( gForce  )       { gForce->Stop();   delete
gForce;   gForce      = NULL; }
          if ( gBall   )  { gBall->Stop();          delete
gBall;    gBall       = NULL; }
          if ( gSpring )       { gSpring->Stop();  delete
gSpring;  gSpring = NULL; }
          if ( gNerf   )  { gNerf->Stop();          delete
gNerf;    gNerf       = NULL; }
          if ( gPop1                )           { gPop1->Stop();
          delete gPop1;       gPop1        = NULL; }
          if ( gPop2                )           { gPop2->Stop();
          delete gPop2;       gPop2        = NULL; }
          if ( gMouse  )            {
                    delete gMouse;      gMouse   = NULL; }
          return TRUE;
}

void FeelEndAllEffects( void )
{
          if ( gForce  )       gForce->Stop();
          if ( gBall   )       gBall->Stop();
          if ( gSpring )       gSpring->Stop();
          if ( gNerf   )       gNerf->Stop();
          if ( gPop1            )         gPop1->Stop();
          if ( gPop2            )         gPop2->Stop();
}

long FeelBeginBall( void )
{
          if ( gBall )
          {   return gBall->Start();
          }
          return 0;
}

long FeelEndBall( void )
{
          if ( gBall )
                    return gBall->Stop();
          return 0;
}
long FeelChangeBallLocation( long left, long top )
{
        if ( gBall )
        {
                RECT r;
                r.top    = top;
                r.left   = left;
                r.right  = left + BALL_IMAGE_WIDTH;
                r.bottom = top  + BALL_IMAGE_HEIGHT;
                return gBall->SetRect( &r );
        }
        return 0;
}

long FeelBeginForce( long Xdir, long Ydir, long Mag )
{
        if ( gForce )
        {
                gForce->ChangeParameters(
                    Xdir,
                         Ydir,
                    FORCE_EFFECT_DONT_CHANGE,
                    Mag
                );
                return gForce->Start();
        }
        return 0;
}

long FeelEndForce( void )
{
        if ( gForce )
                return gForce->Stop();
        return 0;
}

long FeelBeginSpring( long top )
{
        if ( gSpring )
        {
                POINT pt = {0,top};
                gSpring->ChangeParameters(
                        pt
                );
                return gSpring->Start();
        }
        return 0;
}

long FeelEndSpring( void )
{
        if ( gSpring )
                return gSpring->Stop();
        return 0;
}

long FeelSetSpring( long springK )
{
        if ( gSpring )
                return gSpring->ChangeParameters(

        FORCE_EFFECT_DONT_CHANGE_POINT,
                        springK,
                        FORCE_EFFECT_DONT_CHANGE,
                        FORCE_EFFECT_DONT_CHANGE,
                        FORCE_EFFECT_DONT_CHANGE
                );
        return 0;
}

long FeelBeginNerf( void )
{
        if ( gNerf )
        {    return gNerf->Start();
        }
        return 0;
}

long FeelEndNerf( void )
{
        if ( gNerf )
                return gNerf->Stop();
        return 0;
}

long FeelChangeNerfRect( long left, long top, long width,
long height )
{
        if ( gNerf )
        {
                RECT r;
                r.top    = top;
                r.left   = left;
                r.right  = left + width;
                r.bottom = top  + height;
                return gNerf->SetRect( &r );
        }
        return 0;
}

long FeelPop( void )
{
        if ( gPop1 )
                gPop1->Start();
        if ( gPop2 )
                gPop2->Start();
        return 1;
}
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by DynamicControl.rc
//
#define IDS_DYNAMICCONTROL                  1
#define IDS_DYNAMICCONTROL_PPG              2

#define IDS_DYNAMICCONTROL_PPG_CAPTION    200

#define IDD_PROPPAGE_DYNAMICCONTROL       200

#define IDD_ABOUTBOX_DYNAMICCONTROL         1

#define IDB_DYNAMICCONTROL                  1

#define IDI_ABOUTDLL                       1

#define _APS_NEXT_RESOURCE_VALUE           201
#define _APS_NEXT_CONTROL_VALUE            201
#define _APS_NEXT_SYMED_VALUE              101
#define _APS_NEXT_COMMAND_VALUE            32768
```

StdAfx.h

```
#if
!defined(AFX_STDAFX_H__EC296EEA_836C_11D1_A868_0060083A2742_
_INCLUDED_)
#define
AFX_STDAFX_H__EC296EEA_836C_11D1_A868_0060083A2742__INCLUDED
_

#if _MSC_VER >= 1000

#pragma once
#endif // _MSC_VER >= 1000

// stdafx.h : include file for standard system include
files,
//      or project specific include files that are used
frequently,
//      but are changed infrequently

#define VC_EXTRALEAN      // Exclude rarely-used stuff from
Windows headers

#include <afxctl.h>         // MFC support for ActiveX
Controls

// Delete the two includes below if you do not wish to use
the MFC
//  database classes
#include <afxdb.h>                      // MFC database
classes
#include <afxdao.h>                     // MFC DAO database
classes

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional
declarations immediately before the previous line.

#endif //
!defined(AFX_STDAFX_H__EC296EEA_836C_11D1_A868_0060083A2742_
_INCLUDED_)
```

StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard
includes
// stdafx.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division

Application deficiencies found during scanning:

□ Page(s)____________of____________________________________ were not present for scanning. (Document title)

□ Page(s)____________of____________________________________were not present for scanning. (Document title)

□ ***Scanned copy is best available.*** Page 55 - 88 part of Specificat- ions are Appendix.